#15

Table of Contents

MEMO

DATE: December 14, 1999

RE: Dickens U.S. Patent 5806063

Summary

All of the claims are anticipated by at least one of the Ohms, IBM or Shaughnessy references. Ohms, an uncited statutory bar, invalidates all the claims, either alone or in combination with other art.

Errors Leading to issuance of Dickens

1. Failure to uncover Ohms (1986 IBM publication)

2. Failure to rely on Shaughnessy (filed Nov. 21, 1994)

3. Withdrew rejection on IBM (published Oct. 1995) in light of defective 131 Declaration

   a. Declaration defective for failure of evidence in support of conception[1]

   b. Declaration defective for inadequate evidence of reduction to practice

4. Claims were allowed based on subject matter which was NOT in the claims.[2]

---

[1] 'Conception must be proved by corroborating evidence which shows that the inventor disclosed to others his "complete thought expesssed in such clear terms as to enable those skilled in the art" to make the invention.'... However, 'there is no final single formula that must be followed in proving corroboration.... Rather, the sufficiency of corroboration evidence is determined by the "rule of reason."'...Accordingly, a tribunal must make a reasonable analysis of all of the pertinent evidence to determine whether the inventor's testimony is credible.... The tribunal must also bear in mind the purpose of corroboration, which is to prevent fraud, by providing independent confirmation of the inventor's testimony.". Kridl v. McCormick, 105 F.3d 1446, 1449-1450, 41 USPQ2d 1686, 1689 (Fed. Cir. 1997)

[2] .In the reasons for allowance, the examiner indicated that the claims were allowed because the prior art:

> "Does not anticipate nor suggest the set of limitations of the claims, comprising the threshold year digits as used to determine a pair of century digits to be used for computation, but without enlarging the number of date digits of the database" (emphasis added).

There is, however, nothing in the claims to suggest there is no enlargement of the number of date digits of the database.

The major references relied on are:

Ohms, "Computer Processing of Dates Outside the 20th Century", found in the IBM Systems Journal, vol. 25, #2, 1986, (uncited – attachment 11);

Shaughnessy patent 5630118 (cited – attachment 8) and

"The Year 2000 and 2-Digit Dates", IBM, October 1995 (cited – attachment 9).

The Dickens Patent

The patent was issued on September 8, 1998 on the basis of an application filed on October 3, 1996. As filed, the specification comprised five typed pages (attachment 2) and referred to two figures of drawing (attachment 3). The application as filed also included fifteen method claims including independent method claims 1 and 11 (attachments 4 and 5 respectively).

The specification describes the Y2K problem; i.e., the ambiguity raised in attempting to interpret the representation of a year specified in two digits in the context of a century boundary. The specification proposes a preferred format of YYMMDD (Y represents a year digit, M represents a month digit, and D represents a day digit). Given this date format, the patent describes the solution as follows:

> "A 10-decade window with a Y_AY_B value for the first year of the 10-decade window is selected, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database. A century designator C_1C_2 is determined for each date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B. Each date in the database is formatted with the values C_1C_2, Y_1Y_2, M_1M_2, and D_1D_2."

In other words, given a data base or collection of data which is restricted to a date range of no more than 100 years, any two digit year can be properly located relative to the century boundary by selecting a pivot year (sometimes also called a base year) which is no larger than the earliest date in the data base. To locate any year we merely compare the year to the pivot year. If the year is smaller than the pivot year then the date must be beyond the century boundary, i.e. for Y2K the year is in the 21st century. This follows because we chose the pivot year to be as early as the earliest year in the collection. Once we determine that the year is smaller, then by definition the year must be beyond the century boundary. Conversely, if the year is greater than or equal to the pivot year, then the year is determined to be in the earlier century, i.e., the 20th century. The result again follows from the manner in which we chose the pivot year.

For example assume the data base has dates from 1967 on. We select a pivot year of 66 (we could have selected any year less than 66 or even 67, but no more than 67). Assume we retrieve 85 as a year in question. Since 85 is larger than 66 we determine that 85 refers to 1985

and not to 2085. Now assume we retrieve 42. We determine that 42 is less than 66 and so determine that 42 refers to 2042 and not to 1942.

That is windowing. It is claimed by Dickens. Significantly, however, it is also described by the prior art Ohms, IBM, and Millennium publications as well as the Shaughnessy patent.

Prosecution History

On November 17, 1997 the examiner issued an action rejecting all 15 claims. The art cited by the examiner included the Shaughnessy patent 5,630,118 (attachment 8) and a May 1996 edition of an IBM publication entitled "The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation". The examiner rejected claims 1-15 as being anticipated by the IBM publication.

The response to the rejection included an IDS identifying an October 1995 edition of the IBM publication (attachment 9) relied on in the office Action. In addition, the response included a declaration under rule 131, intended to antedate the October 1995 date of the IBM publication identified in the IDS. The declaration alleged a reduction to practice of the invention in April 1996 as well as conception of the invention at some unidentified date prior to October 1995, coupled with diligence toward the reduction to practice in April 1996. The declaration is found as attachment 6 and the exhibits to the declaration are found in attachment 6A. There was no attempt to distinguish the claims from the applied reference or any other reference.

Several days later (April 2), a telephone interview was conducted and as a consequence the applicant filed a supplemental response amending independent claims 1 and 11 as well as claims 4, 6, 8, 10 and 13 in order to overcome a rejection based on section 112. Attachments 4 and 5 show (at the bottom of each page) the amendments to claims 1 and 11. There was no attempt to distinguish the claims from the applied reference or any other reference.

The examiner responded (on April 8) by indicating that claims 1-15 were allowed. The examiner recorded his reasons for allowance. He said:

> "The prior art of record, taking into account the affidavit of the inventor, received 3/24/98, swearing behind the reference of the previous action, does not anticipate nor suggest the set of limitations of the claims, comprising the threshold year digits as used to determine a pair of century digits to be used for computation, but without enlarging the number of date digits of the database". (Emphasis added - Attachment 7).

None of the Dickens claims includes any limitation corresponding to the emphasized language.

The patent issued on September 8.

Prior Art

The art referred to includes:

(1) "The Millennium Journal", July 1995 (attachment 10),
(2) Ohms, "Computer Processing of Dates Outside the 20th Century" found in the IBM Systems Journal, volume 25 number 2, 1986 (attachment 11), and
(3) DeJager, "Doomsday 2000" found in the September 6, 1993 issue of Computerworld (attachment 13).

Subject Matter of the References

The nub of Dickens is that a century designator can be determined from a 2-digit year date (1) if the date range is limited to no more than 100 years, (2) appropriately selecting a pivot year (Y_AY_B) and (3) testing the 2-digit year date against the pivot year to determine the century designator. This description is found in each one of Shaughnessy, IBM, Millennium, and Ohms going back in time (at least in the case of Ohms) to 1986.

Shaughnessy (attachment 8)

Referring to Fig. 2, Shaughnessy first determines the current date, then uses that to determine the end of the 100 year cycle, and from that determines two possible century values. To address Y2K, those values are the 20th and 21st centuries corresponding to the century designator pairs 19 and 20. Fig. 7 illustrates how the century value is assigned. Function 62 compares the date (that is the 2-digit year representation which is being processed) with the end of the 100-year cycle. If the date is less than or equal (see column 7, line 8-10), the 21st century indicator is assigned whereas if the date is greater than the end of cycle date, then the earlier century designator is assigned.

IBM (attachment 9)

The IBM publication on the Y2K subject includes chapter 4 "Reformatting Year-Date Notation". The text provides:

> "The fixed window technique uses a static 100-year interval that generally crosses a century boundary. This technique determines the century of a 2-digit year by comparing the 2-digit year against a window of 100 years. The user specifies the number of years in the past and future relative to a specific year within the 100-year interval."

The text uses an example where the pivot year is 1960 and determines the century for a given date XY by first determining if XY is greater or equal to 60 (the pivot year). If it is, we have a 20th century date, e.g., the century indicator is 19. On the other hand, if XY is less than or equal to 59, e.g., less than 60, then the date is a 21st century date, e.g., the century indicator is 20.

Millennium

The Millennium publication also describes windowing. The text indicates:

"In windowing, the 2-digit years are left alone in the files. A base year is selected (e.g., "50") where every year starting with (or, in some cases, greater than) "50" through "99" is treated as a 1900 date, and any year less than (or equal to) "50" is treated as 2000.

The text also indicates that "Sorts require an exit.....".

Ohms

Ohms also addresses the Y2K problem, and it does so at a much earlier time than the other references. Ohms notes, at page 248:

"However, it may be necessary to provide a conversion function that receives a definition of the implied century as a parameter. An excellent way to do this unambiguously is to specify a year as the desired starting point of a 100-year range. For example, if the starting year for the range is specified as 1925, dates with year digits of 25 through 99 would be between 1925 and 1999, and dates with year digits of 00 through 24 would lie between 2000 and 2024."

DeJager

DeJager describes, from the vantage of 1993, the need for paying attention to Y2K problems. The pertinent portion of DeJager indicates that failure to pay attention to Y2K can have an impact on sorting functions.

Attached and forming a part of this memo is a set of claim tables (Tables A-M) and attachments 1-16 which are identified as follows

1. The Dickens Patent
2. The specification of the application, as filed
3. The set of drawings from the application, as filed
4. Independent Claim 1, as filed, also showing a later amendment
5. Independent Claim 11, as filed, also showing a later claim amendment
6. The Declaration of Inventor filed under Rule 131

6.A The Exhibits to The Declaration of the Inventor under Rule 131

7. The Examiner's Reasons for Allowance
8. The cited Shaughnessy reference
9. The cited IBM reference (October 1995 edition)
10. The uncited Millennium Journal reference

11. The uncited Ohms reference
12. The uncited Roberts reference
13. The uncited DeJager reference
14. Common Lisp, The Language (1984)
15. Computer Software Corp. Web page on DateServer2000
16. ISO 2014 Numeric Calendar Dates 1976

Table A

	Ohms, Computer Processing of dates outside the twentieth century, 1986
1. A method of processing symbolic representations of dates stored in a database, comprising the steps of	Ohms describes a "date processing method" (p. 244
providing a database with symbolic representations of dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;	this is a classic short Gregorian date, see p. 247, the conversion function described at p. 248 works with any format which includes 2 digit year. Ohms describes the 100 year (i.e., 10-decade) period of time (p.249)
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	p. 248, right hand column,-specify a year as the desired starting point of the range - this is Y.sub.A Y.sub.B , which is no later than any year date in the data base
determining a century designator C.sub.1 C.sub.2 for each symbolic representation of a date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ; and	the century designation is determined by comparing the year date (Y.sub.1 Y.sub.2) with Y.sub.A Y.sub.B , if the year, Y.sub.1 Y.sub.2, is greater then the century is the earlier one and vice versa, see p. 248
reformatting the symbolic representation of the date with the values C.sub.1 C.sub.2, Y.sub.1 Y.sub.2, M.sub.1 M.sub.2 , and D.sub.1 D.sub.2 to facilitate further processing of the dates.	the "implied century" (see p. 248, right hand column) is the

Table B

	Shaughnessy US Patent 5630118
1. A method of processing symbolic representations of dates stored in a database, comprising the steps of	The disclosed software assigns a century value to a two digit year date (7/6+), which is processing of symbolic dates
providing a database with symbolic representations of dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;	One format which can form an input is YYMMDD, see Date Type "B" in the appendix at col.18, This processing is limited to dates which span 100 years, see below
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	software "determine[s] end of current 100 year cycle", step 16, fig. 2, 3 or 4, as the "end" of the 100 year range, the "end" year is one less than the beginning (if "37" is the last year of a 100 year period, "38" is the first year of the same period), the "end" year is no later than any date in the data base as claimed
determining a century designator C.sub.1 C.sub.2 for each symbolic representation of a date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ; and	the century designator is determined by comparing two digit representation to the end of the 100 year cycle date, if the year being processed is greater, then the earlier century value is assigned and vice versa; (col. 7, lines 5-15)
reformatting the symbolic representation of the date with the values C.sub.1 C.sub.2, Y.sub.1 Y.sub.2, M.sub.1 M.sub.2 , and D.sub.1 D.sub.2 to facilitate further processing of the dates.	the reformatting is described at 2/30-32; 6/57-

Table C

1. A method of processing symbolic representations of dates stored in a database, comprising the steps of	IBM, THE YEAR 2000 AND 2-DIGIT DATES, Oct. 1995 The Chapter heading, "reformatting year-date notation" relates to processing symbolic representations of dates
providing a database with symbolic representations of dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;	in the "fixed window" section, the reference to "2 digit year" is a reference to the date format of this claim the text cautions that the procedure is limited to 100 year ranges, i.e., a 10-decade period of time
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	the text refers to the 100 year window, and as shown the selection of "60" as the value for Y.sub.A Y.sub.B is indeed no later than the earliest year date being processed
determining a century designator C.sub.1 C.sub.2 for each symbolic representation of a date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ; and	the determination of the century designator is exactly as claimed, (the notation in the text, Y.sub.A Y.sub.B $\geq$60 or Y.sub.A Y.sub.B $\leq$ 59 is identical to Y.sub.A Y.sub.B < 60 which is claimed)
reformatting the symbolic representation of the date with the values C.sub.1 C.sub.2, Y.sub.1 Y.sub.2, M.sub.1 M.sub.2 , and D.sub.1 D.sub.2 to facilitate further processing of the dates.	use of the century designator in reformatting year data is indicated by the very chapter heading

Table D

2. The method of claim 1, wherein the 10-decade window includes the decade beginning in the year 2000.	Ohms, IBM and Shaughnessy all are directed to Y2K and by definition propose a window which includes the year 2000
3. The method of claim 2, wherein the step of determining includes the step of determining the first value as 20 and the second value as 19.	Ohms, IBM and Shaughnessy all are directed to Y2K and by definition propose century designators 19 and 20
4. The method of claim 1, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates.	Shaughnessy suggest a further sorting operation or an operation equivalent to sorting. Shaughnessy describes date comparisons (col 4, lines 37-62, col. 8, line 33-col. 12, line 19), IBM has cautions (p. 4-3) related to the use of the data as a sequence for indexing
5. The method of claim 1, wherein the step of reformatting includes the step of reformatting each symbolic representation of a date into the format C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2.	Shaughnessy uses this format, see 6/58
6. The method of claim 5, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates using a numerical-order sort.	Shaughnessy teaches using the reformatted data for a date comparisons, see (col 4, lines 37-62, col. 8, line 33-col. 12, line 19). This is equivalent to sorting
7. The method of claim 1, wherein the step of providing a database includes the step of converting pre-existing date information having a different format into the format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator and Y.sub.1 Y.sub.2 is the numerical year designator.	Shaughnessy teaches a host of date formats (appendix in col. 18) and indicates they can be converted to the format using M, D and Y variables as claimed, see col. 8, lines 18-27.

Table E

	Ohms, Computer Processing of dates outside the twentieth century, 1986 and DeJager, "Doomsday", 1993
4. The method of claim 1, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates.	The subject matter of Ohms has already been described. DeJager, in his 1993 Y2K wakeup call entitled "DOOMSDAY" points out (see p. 108) that the failure to properly handle Y2K will produce sorting errors. Since both references address Y2K, Ohms describes one solution while DeJager underlines the importance of any solution, it would be well within ordinary skill, considering both references, to realize that correctly handling Y2K will enhance sorting functions as claimed.

Table F

8. The method of claim 1, wherein the step of selecting includes the step of selecting Y.sub.A Y.sub.B such that Y.sub.B is 0 (zero).	Both IBM and the Millennium Journal has an example of the pivot year of "60" or "50" as claimed - See the text on the other 103 factors relative to the use of the Millennium Journal
9. The method of claim 1, including an additional step, after the step of reformatting, of storing the symbolic representation of dates and their associated information back into the database.	Ohms and IBM teach that storing the reformatted dates can be done, see pp 248-9 of Ohms and the discussion of compressed code solutions in IBM.
10. The method of claim 9, including the additional step, after the step of reformatting, of manipulating information in the database having the reformatted date information therein.	data bases are used for accessing and using the stored information

Table G

	Shaughnessy US Patent 5630118
11. A method of processing dates in a database, comprising the steps of	The disclosed software assigns a century value to a two digit year date (7/6+) which is processing of dates
providing a database with dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;	One format which can form an input is YYMMDD, see Date Type "B" in the appendix at col.18, This processing is limited to dates which span 100 years, i.e., a 10-decade period (see below)
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window, Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	software "determine[s] end of current 100 year cycle", step 16, fig. 2, 3 or 4, as the "end" of the current cycle it is also at least as "early" as any date in the database, for example, if "37" were the end of the 100 year period, then "38" would be the beginning and either "37" or"38" would be no later than any date in the range, as claimed,
determining a century designator C.sub.1 C.sub.2 for each date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ;	a century designator is determined by comparing a two digit year representation to the end of the 100 year cycle date, if it is greater, then the earlier century value is assigned and vice versa; (col. 7, lines 8-13)
reformatting each date in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2 to facilitate further processing of the dates; and	the reformatting is described at 2/30-32 and 6/57-
sorting the dates in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2.	sorting is a well known operation for the date comparison described at 1/26 and at column 8

Table H

	IBM, THE YEAR 2000 AND 2-DIGIT DATES, Oct. 1995
11. A method of processing dates in a database, comprising the steps of	The Chapter heading, "reformatting year-date notation" relates to processing dates
providing a database with dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;	in the "fixed window" section, the reference to a "2 digit year" is a reference to the date format of this claim the text cautions that the procedure is limited to 100 year ranges, i.e., a 10-decade period
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window, Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	the text refers to the 100 year window, and as shown selection of 60 for the Y.sub.A Y.sub.B is indeed no later than the earliest year date being processed
determining a century designator C.sub.1 C.sub.2 for each date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ;	the determination of the century designator is exactly as claimed, (the notation in the text, Y.sub.A Y.sub.B $\geq$60 or Y.sub.A Y.sub.B $\leq$ 59 is identical to Y.sub.A Y.sub.B < 60 which is claimed)
reformatting each date in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2 to facilitate further processing of the dates; and	use of the century designator in reformatting year data is indicated by the very chapter heading
sorting the dates in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2.	reference to collating sequence support and indexing sequence support suggest use of reformatted data in operations like sorting

Table I

12. The method of claim 11, wherein the step of providing a database includes the step of	
converting pre-existing date information having a different format into the format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator and Y.sub.1 Y.sub.2 is the numerical year designator.	Shaughnessy teaches a host of date formats which can be converted to the format using M, D and Y variables as claimed, see the appendix at column 18, the conversion among these formats is taught at column 8, lines 18-27
13. The method of claim 11, wherein the step of selecting includes the step of selecting Y.sub.A Y.sub.B such that Y.sub.B is 0 (zero).	Both IBM and the Millennium Journal have examples of the pivot year of "60" or "50, both of which have a "0" unit digits
14. The method of claim 11, including an additional step, after the step of sorting, of storing the sorted dates and their associated information back into the database.	IBM, teaches that storing the reformatted dates can be done, see the discussion of compressed code solutions in IBM.
15. The method of claim 14, including the additional step, after the step of sorting, of manipulating information in the database having the reformatted date therein.	data bases are used for accessing and using the stored information

Table J

	Ohms, Computer Processing of dates outside the twentieth century, 1986 and The Millennium Journal , July 1995
5. The method of claim 1, wherein the step of reformatting includes the step of reformatting each symbolic representation of a date into the format C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2.	The Millennium Journal, at p. 4 describes the claimed format in the context of a Y2K solution. This teaching would have made it obvious at the time to use this format after using the Ohms procedure to determine the proper century.
6. The method of claim 5, including an additional step, after the step of reformatting, of sorting the symbolic representations of dates using a numerical-order sort.	The Millennium Journal, at p. 3, refers to sorting of dates in connection with Y2K procedures. This teaching would have made it obvious at the to sort data after Y2K.
7. The method of claim 1, wherein the step of providing a database includes the step of converting pre-existing date information having a different format into the format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator and Y.sub.1 Y.sub.2 is the numerical year designator.	Ohms describes a number of date format conversions as well as the use of the specific YYMMDD format.

Table K

	Ohms, Computer Processing of dates outside the twentieth century, 1986 and The Millennium Journal , July 1995
14. The method of claim 11, including an additional step, after the step of sorting, of storing the sorted dates and their associated information back into the database.	Ohms teaches that storing the reformatted dates can be done, see the discussion at p. 248-9.
15. The method of claim 14, including the additional step, after the step of sorting, of manipulating information in the database having the reformatted date therein.	data bases are used for accessing and using the stored information

Table L

	Ohms, Computer Processing of dates outside the twentieth century, 1986 and The Millennium Journal , July 1995
11. A method of processing dates in a database, comprising the steps of	Ohms describes a "date processing method" (p. 244, 5
providing a database with dates stored therein according to a format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator, and Y.sub.1 Y.sub.2 is the numerical year designator, all of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;	this is a classic short Gregorian date, see p. 247, the conversion function described at p. 248 works with any format which includes 2 digit year representations as is this one. Ohms describes the 100 year, i.e., 10-decade period (p.249)
selecting a 10-decade window with a Y.sub.A Y.sub.B value for the first decade of the window, Y.sub.A Y.sub.B being no later than the earliest Y.sub.1 Y.sub.2 year designator in the database;	p. 248, right hand column,- specify a year as the desired starting point of the range - this is Y.sub.A Y.sub.B, which is no later than any year date in the data base
determining a century designator C.sub.1 C.sub.2 for each date in the database, C.sub.1 C.sub.2 having a first value if Y.sub.1 Y.sub.2 is less than Y.sub.A Y.sub.B and having a second value if Y.sub.1 Y.sub.2 is equal to or greater than Y.sub.A Y.sub.B ;	the century designation is determined by comparing the year date (Y.sub.1 Y.sub.2) with Y.sub.A Y.sub.B, if the year, Y.sub.1 Y.sub.2, is greater then the century is the earlier one and vice versa, see p. 248
reformatting each date in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2 to facilitate further processing of the dates; and	the Millennium Journal does describe this format in the context of Y2K procedures - consequently it would have been obvious at the time to have used this format with the century information when acquired as described by Ohms
sorting the dates in the form C.sub.1 C.sub.2 Y.sub.1 Y.sub.2 M.sub.1 M.sub.2 D.sub.1 D.sub.2.	the Millennium Journal also indicates, p. 3, that sorting can be accomplished in connection with Y2K date correction

Table M

	Ohms, Computer Processing of dates outside the twentieth century, 1986 and The Millennium Journal , July 1995
12. The method of claim 11, wherein the step of providing a database includes the step of	
converting pre-existing date information having a different format into the format wherein M.sub.1 M.sub.2 is the numerical month designator, D.sub.1 D.sub.2 is the numerical day designator and Y.sub.1 Y.sub.2 is the numerical year designator.	Ohms describes a number of date format conversions as well as the use of the specific YYMMDD format.

Specification as Filed

DATE FORMATTING AND SORTING FOR DATES SPANNING THE TURN OF THE CENTURY

BACKGROUND OF THE INVENTION

This invention relates to the manipulation of information in a database, and, in particular, to the determination of dates in a useful form.

Dates are stored as symbolic representations in computer databases in varying formats. For example, a date may be represented in the numerical representation MM/DD/YY, where MM is a two-digit month designator, DD is a two-digit day designator, and YY is a two-digit year designator (the last two digits of the year). Thus, December 15, 1993 is designated as 12/15/93. A date may also be represented in an alphanumeric form MMM/DD/YY, where MMM is an alphabetic month designator (e.g., DEC for December), and DD and YY are the same as in the numerical form. December 15, 1993 is represented in this format as DEC/15/93.

Such approaches for the representation of dates have worked well since the advent of computer databases, which has occurred in the twentieth century. Dates may be sorted in chronological order using the numerical representations. However, with the turn of the century at January 1, 2000, the representation and utilization of dates becomes more complex. Using the numerical form above, December 15, 2000 is represented as 12/15/00. If a numerical sort is performed on 12/15/93 and 12/15/00, the later date 12/15/00 sorts as the first-occurring date, an incorrect result.

Sets of dates spanning the turn of the century and associated with past, current, and future activities are now stored in many databases. When stored in the conventional formats discussed above, those dates will not readily be used and numerically sorted in chronological order. They may be manually converted to a more usable form in the sense that programs may be written to perform conversions, manipulations, and sorting. However, these programs typically require additional data fields for storage, which may be objectionable in some circumstances.

There is a need for an improved approach to the representation and utilization of dates in databases, and for converting the existing dates in databases to a more usable form. The present invention fulfills this need, and further provides related advantages.

SUMMARY OF THE INVENTION

The present invention provides an approach to the representation and utilization of dates stored symbolically in databases. Existing symbolic date representations are converted to a more useful form of symbolic date representations without the addition of new data fields, and in a manner that is performed automatically by the computer and requires no user input. The approach of the invention permits direct numerical sorting of dates.

In accordance with the invention, a method of processing dates stored in a database comprises the steps of providing a database with dates stored therein according to a format wherein M_1M_2 is the numerical month designator, D_1D_2 is the numerical day designator, and Y_1Y_2 is the numerical year designator, all of the dates falling within a 10-decade period of time. A 10-decade window with a Y_AY_B value for the first year of the ten-decade window is selected, Y_AY_B being no later than the earliest Y_1Y_2 year designator in the database. A century designator C_1C_2 is determined for each date in the database, C_1C_2 having a first value if Y_1Y_2 is less than Y_AY_B and having a second value if Y_1Y_2 is equal to or greater than Y_AY_B. Each date in the database is formatted with the values C_1C_2, Y_1Y_2, M_1M_2, and D_1D_2.

In the case of most practical interest, the 10-decade period of time spans the year 2000 and begins with a year in which the second digit (Y_B in Y_AY_B) is 0 (zero). For any 10-decade period including the year 2000, if the decade designator Y_1 of the date in the database is numerically less than the decade designator Y_A of the first decade of the 10-decade period of time, the century designator C_1C_2 is "20". If Y_1 is equal to or greater than Y_A, C_1C_2 is "19". Dates in databases spanning more than 10 decades are not handled by this approach, but it is not expected that

this limitation will be significant for most commercial and industrial databases.

This approach works particularly well if the dates are represented in the format $C_1C_2Y_1Y_2M_1M_2D_1D_2$. The date Dec. 15, 2000 is represented in this format as 20001215, for example. Dates represented in this format may be directly sorted numerically by fast sorting techniques, and thereafter stored back in the database.

The present invention thus provides an efficient approach to converting and utilizing symbolic date representations in databases, which allows automatic processing of dates ranging from before to after the year 2000. The large number of dates represented in some databases may thereby be readily processed and utilized. Other features and advantages of the present invention will be apparent from the following more detailed description of the preferred embodiment, taken in conjunction with the accompanying drawings, which illustrate, by way of example, the principles of the invention. The scope of the invention is not, however, limited to this preferred embodiment.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic representation of a computer database with date information therein; and

Figure 2 is a block flow diagram of a preferred approach for practicing the approach of the invention.

DETAILED DESCRIPTION OF THE INVENTION

Figure 1 schematically depicts a computer 20 having a read-only or random-access memory 22, a mass-storage device 23, and a central processing unit 24 therein. Stored in the memory 22 or on the mass-storage device 23 is a database 26. The database includes information in the form of symbolic representations of dates and associated information such as events occurring on the respective dates. In a conventional approach, the dates are stored in a format such as $M_1M_2/D_1D_2/Y_1Y_2$ format. M indicates month information, D day information, and Y year information,

with the subscript 1 or 2 indicating the first or second digit of the designator, respectively. December 15, 1993 is stored as 12/15/93 or 12-15-93, and December 15, 2000 is stored as 12/15/00 or 12-15-00, for example. If a numerical sort is performed on these dates, 12/15/00 will sort chronologically prior to 12/15/93.

Figure 2 illustrates the approach of the invention. The computer database 26 is provided, numeral 30, having symbolic representations of dates stored therein. In some cases, the dates will be represented as discussed in the preceding paragraph. In other cases, an alphanumeric designator is used. In that approach, each date is stored as $M_aM_bM_c/D_1D_2/Y_1Y_2$ format, where $M_aM_bM_c$ is an alphabetical symbol such as JAN for January, FEB for February, etc. In that case, the month designator $M_aM_bM_c$ is first converted to the numerical form M_1M_2 by converting JAN to "01", FEB to "02", etc.

A 10-decade window is selected, numeral 32. That is, it is necessary that all dates in the database will be within some period of 10 decades, or 100 years. This limitation poses little problem for most industrial and commercial databases. The window may be arbitrarily selected. For example, the decade could begin with the 1950's and end with the 2040's, or it could begin with the 1980's and end with the 2070's. The 10-decade window will normally include some decades from the prior century and some from the new century.

The first year of the 10-decade window is represented by Y_AY_B. In a commonly utilized application, Y_B is 0 (zero), although the invention is not limited to this case. That is, the 1950's first decade would be represented by Y_A0 of "50", and the 1980's first decade would be represented by Y_A0 of "80". For this case, a century designator C_1C_2 for a date is determined, numeral 34, by comparing the value of Y_1, the first digit of the year designator for the date, with Y_A, the first digit of the first decade of the 10-decade window. C_1C_2 is assigned a first value if Y_1 is less than Y_A and a second value if Y_1 is equal to or greater than Y_A.

In the case of most interest, the 10-decade window includes decades earlier than the year 2000 and decades later than the year 2000, and Y_B is zero C_1C_2 is assigned "20" if Y_1 is less than Y_A and is assigned "19" if Y_1 is equal to or greater than Y_A. In that case and for example, if Y_A is 5, meaning that the decade

beginning in 1950 was selected as the first decade of the 10-decade window, and if Y_1Y_2 is "43", the century designator C_1C_2 is "20", indicating that the year in question in the database is 2043. On the other hand, if Y_1Y_2 is "63", the century designator C_1C_2 is "19", indicating that the year in question in the database is 1963. This selection process is performed in a completely automated fashion by the computer, without human input other than to select the starting date of the 10-decade window.

The symbolic representations of the dates in the database are reformatted with the values C_1C_2, Y_1Y_2, M_1M_2, and D_1D_2, numeral 36 of Figure 2. In one case that produces particularly advantageous results for many operations, such as chronological date sorting, the date is represented in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$. For example, the date 12/15/93 (December 15, 1993) is represented as 19931215 and the date 12/15/00 (December 15, 2000) as 20001215. A straightforward numerical sort of date data fields expressed in this form produces an accurate chronological ordering.

Once the symbolic representations of the dates are reformatted according to the procedures set forth above, the date information may be sorted, numeral 38, or otherwise manipulated, numeral 40, together with the entries associated with the dates. Such manipulation may include handling of data associated with the dates, storing the dates and associated information back in the data base, or other processes.

The approach of the invention has been implemented in a computer program, a copy of which is attached as Exhibit A. This program converts dates both before and after the year 2000.

The present invention provides an effective technique for reformatting symbolic representations of date information that is rapid and automated, and yields new symbolic representations of date information that are particularly amenable to further processing. Although a particular embodiment of the invention has been described in detail for purposes of illustration, various modifications and enhancements may be made without departing from the spirit and scope of the invention. Accordingly, the invention is not to be limited except as by the appended claims.

Drawings as Filed

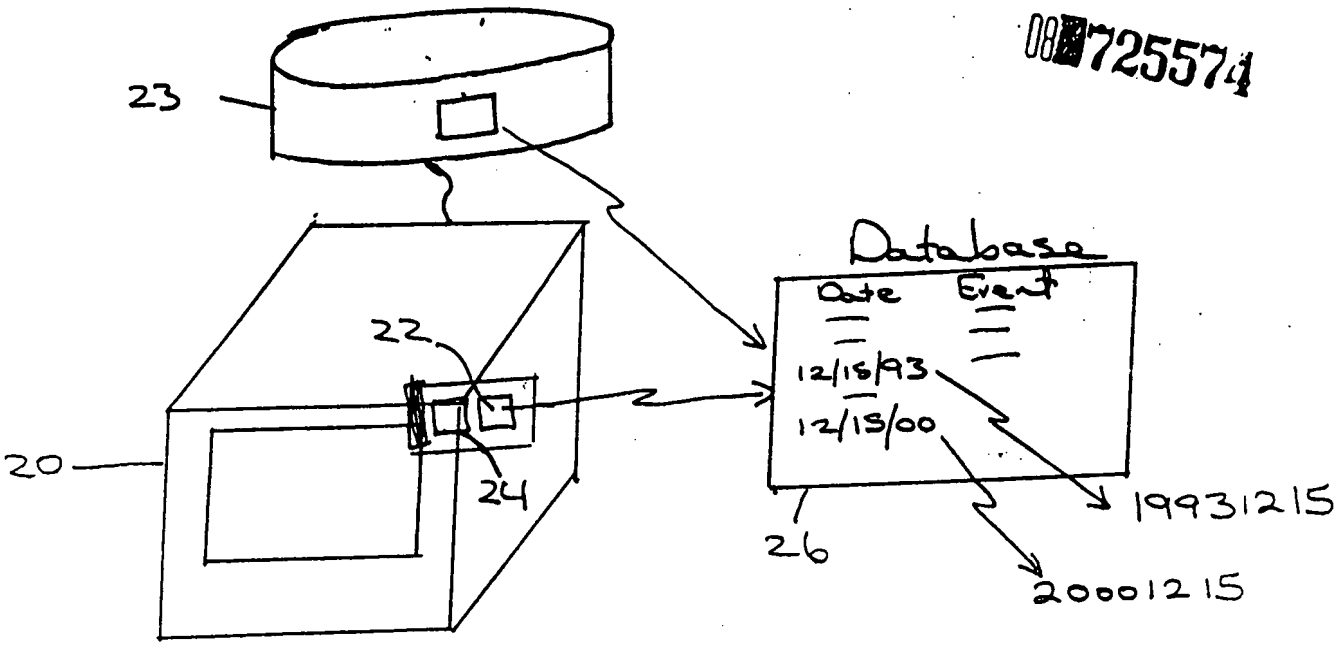

23
22
20
24
Database
Date Event
12/15/93
12/15/00
19931215
20001215
26

Fig 1

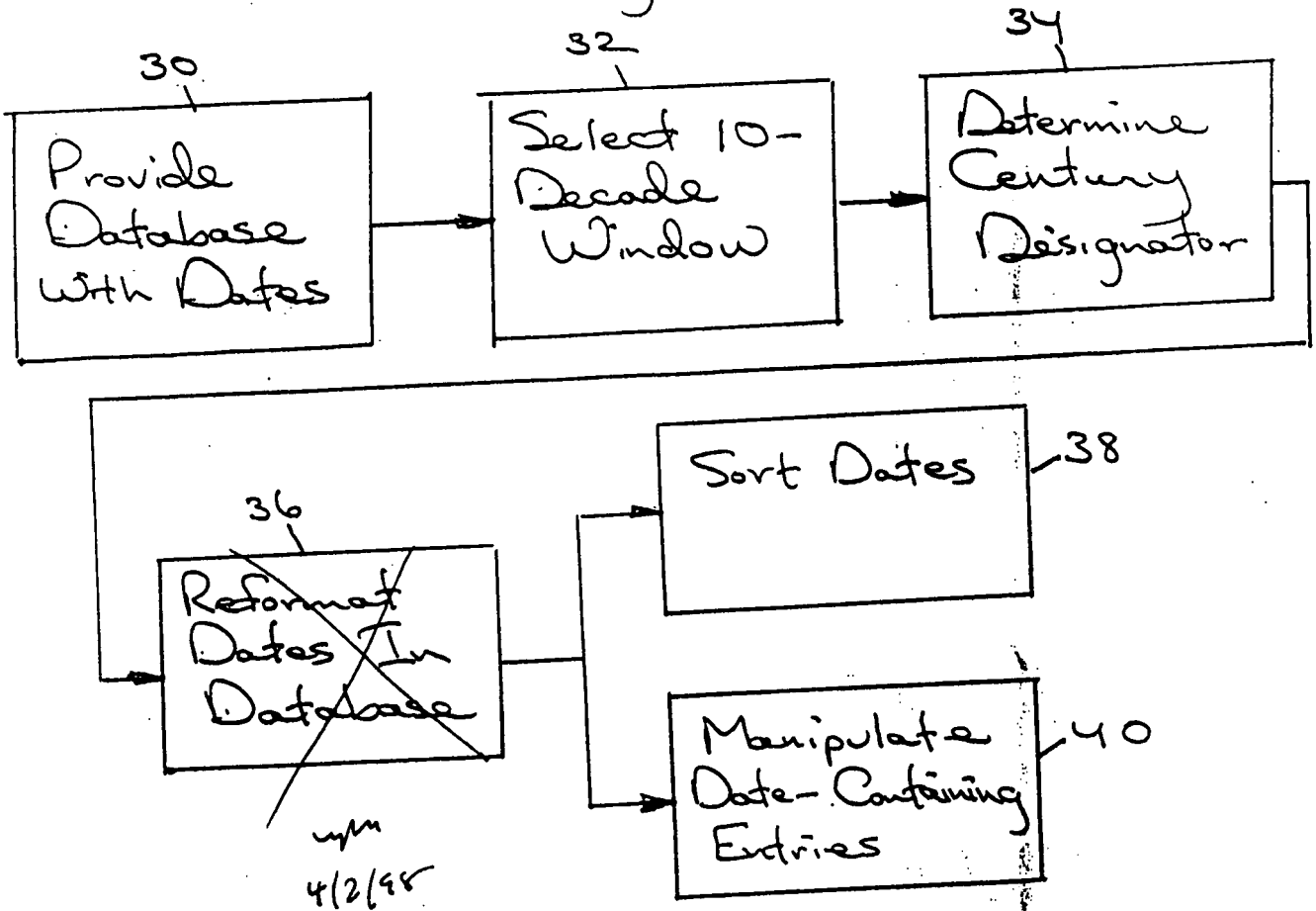

30
Provide Database with Dates
32
Select 10-Decade Window
34
Determine Century Designator
38
Sort Dates
36
Reformat Dates In Database
40
Manipulate Date-Containing Entries
wpm
4/2/98

Fig 2

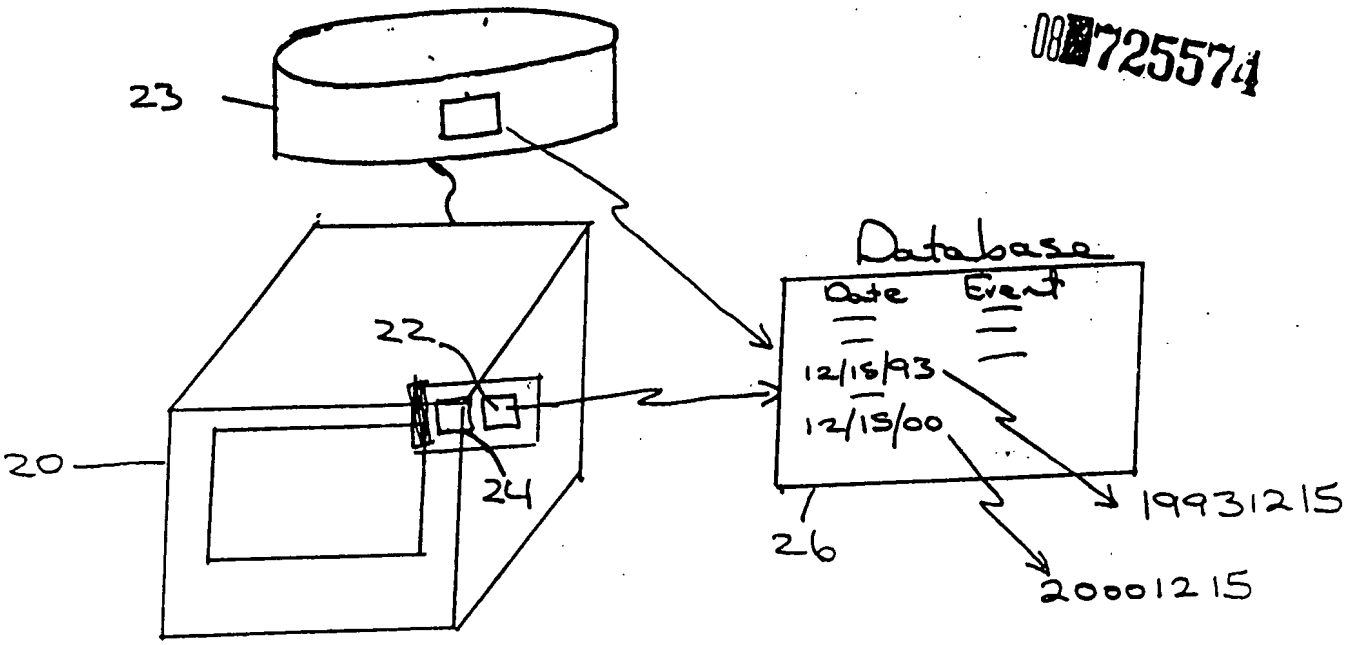

23
22
24
20
Database
Date Event
12/15/93
12/15/00
19931215
20001215
26

Fig 1

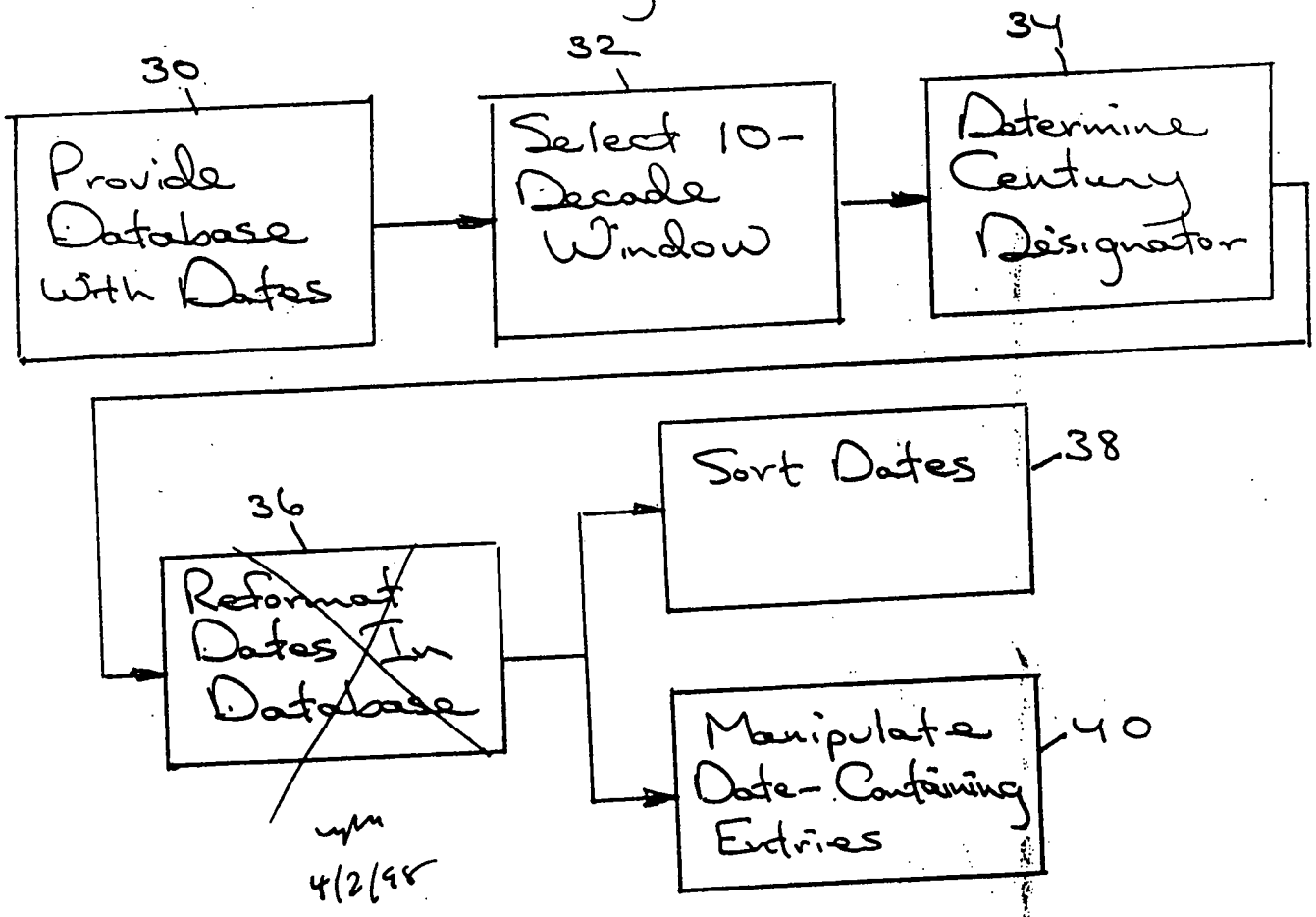

30
Provide Database With Dates
32
Select 10-Decade Window
34
Determine Century Designator
Sort Dates
38
36
Reformat Dates In Database
Manipulate Date-Containing Entries
40
wpm
4/2/98

Fig 2

Claim 1

Claim 1 as filed

1. A method of processing symbolic representations of dates stored in a database, comprising the steps of

providing a database with symbolic representations of dates stored therein according to a format wherein M_1 M_2 is the numerical month designator, D_1 D_2 is the numerical day designator, and Y_1 Y_2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time;

selecting a 10-decade window with a Y_A Y_B value for the first decade of the window Y_A Y_B being no later than the earliest Y_1 Y_2 year designator in the database;

determining a century designator C_1 C_2 for each symbolic representation of a date in the database, C_1 C_2 having a first value if Y_1 Y_2 is less than Y_A Y_B and having a second value if Y_1 Y_2 is equal to or greater than Y_A Y_B ; and

reformatting the symbolic representation of the date in the database with the values C_1 C_2, Y_1 Y_2, M_1 M_2, D_1 D_2.

The last clause of claim 1 is amended as follows:

reformatting the symbolic representation of the date [in the database] with the values C_1 C_2, Y_1 Y_2, M_1 M_2, D_1 D_2 to <u>facilitate further processing of the dates.</u>

Claim 11

Claim 11 as filed

11. A method of processing dates in a database, comprising the steps of

providing a database with symbolic representations of dates stored therein according to a format wherein M_1 M_2 is the numerical month designator, D_1 D_2 is the numerical day designator, and Y_1 Y_2 is the numerical year designator, all of the symbolic representations of dates falling within a 10-decade period of time which includes the decade beginning in the year 2000;

selecting a 10-decade window with a Y_A Y_B value for the first decade of the window Y_A Y_B being no later than the earliest Y_1 Y_2 year designator in the database;

determining a century designator C_1 C_2 for each date in the database, C_1 C_2 having a first value if Y_1 Y_2 is less than Y_A Y_B and having a second value if Y_1 Y_2 is equal to or greater than Y_A Y_B ; and

reformatting each date in the database and sorting the dates in the database using a numerical-order sort.

The last clause of claim 11 is amended as follows:

reformatting each date [in the database] <u>in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$ to facilitate further processing of the dates</u> and
sorting the dates in the database <u>.in the form $C_1C_2Y_1Y_2M_1M_2D_1D_2$.</u>

Declaration of Inventor

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Bruce Dickens
Serial No. 08/725,574
Filed: October 3, 1996
For: DATE FORMATTING AND SORTING FOR DATES SPANNING THE TURN OF THE CENTURY

Group Art Unit: 2771
Examiner: Wayne Amsbury

Assistant Commissioner for Patents
Washington, DC 20231

DECLARATION UNDER 37 C.F.R. § 1.131

Sir:

I, Bruce Dickens, hereby declare and state that:

1. I am the sole inventor of the claimed invention of the above-identified U.S. Patent Application Serial No. 08/725,574 directed to a method of reformatting data stored in a database to overcome the problems arising from the improper recognition and processing of the century designations for dates.

2. I am presently employed by The Boeing Company in Irvine, California, which merged with my previous employer, McDonnell Douglas Corporation, on August 1, 1997. As such, I have worked continuously for first McDonnell Douglas Corporation and now The Boeing Company from February 6, 1988 until the present.

3. During the course of my employment with first McDonnell Douglas Corporation and now The Boeing Company, I have developed software to sort and otherwise process a variety of data. In order to properly process the data, I have had to develop software which reformats the data into substrings, each of which can be separately processed. As shown in Exhibit A, for example, I developed software that reformats a numerical representation of a tool into substrings for subsequent processing. As set forth at line 8 of the code attached at Exhibit A and designated by an "*", the character A represents the designation of different tools stored in a warehouse. For instance, the character A may represent the serial number and make of the part as "SN/Make." As such, the software that I developed and is attached as Exhibit A can reformat the character into two substrings (i.e., SN and Make) such that the tools can be separately sorted or otherwise processed by serial number and make based upon the substrings

"SN" and "Make", respectively. I developed the computer program attached as Exhibit A and named otmout.int;2 at least as early as September 27, 1994, as shown in the directory listing of Exhibit B. In this regard, Exhibit B is a printout of the directory including a listing for the computer program attached as Exhibit A and named "otmout.int;2" that indicates a creation date of September 27, 1994. (See Exhibit B, line 48, designated by an "**").

4. As early as February 1995, I realized that transforming date into substrings could be used to sort and otherwise process dates. For example, most dates are stored in databases as mm/dd/yy representations. I determined that these dates could be represented as substrings by removing the "/"s and reformatting the dates into substrings in the form of mmddyy. These substrings can then be used for sorting by month, day, or year. As such, I developed a computer program, fixmrr.int;2, which performs sorting of dates by reformatting dates into substrings. A printout of fixmrr.int;2 is attached at Exhibit C. As shown at page 5, line 5 of Exhibit C and also designated by an "**", the program reformats a date in the database, MRR(DATE), into a variable ADATE$ that is comprised of substrings. As also shown on page 5, line 11 of Exhibit C and designated by an "***", the program then sorts the dates by using the yy substring of the ADATE$ string. As shown in the directory printout attached as Exhibit D, the creation date of the fixmrr.int;2 program is at least as early as February 6, 1995. (See Exhibit D, line 11, designated by an "**").

5. As demonstrated by Exhibit E, I also developed other types of computer programs that reformatted dates into substrings. In this regard, Exhibit E is a printout of a computer program that determines the number of business days in a year. As can be seen from the output of this program set forth in Exhibit F, the working days of the year have been reformatted as 9998yyyymmdd. (See Exhibit F, col. 1). Further, in reference to Exhibit F, the dates of the year have been reformatted into substrings yymmddyynnn, wherein nnn represents the particular work day of the year. (See Exhibit F, col. 2). As indicated on page 1 of the computer program of

Exhibit E, this program was created at least as early as March 13, 1995.

6. During the development of the computer programs attached as Exhibits C and E and prior to October 1995, I conceived of the claimed method for date formatting and sorting of dates spanning the turn of the century based upon my determination that reformatting dates into substrings could be used to remedy the year 2000 problem. From a date prior to October 1995, I then worked diligently to reduce to practice the claimed invention of date formatting and sorting. For example, once the dates have been parsed into their respective month, day, and year designations, I determined that the yy substring of a date could be tested against the earliest year in a ten decade period to determine whether the date was prior to or after the year 2000. For instance, a date represented by ADATE$ could be reformatted as mmddyy. The yy substring could then be compared to the earliest year in a ten decade period to determine whether the date was in the twentieth or twenty-first century. If the year designation for a particular date in the database is less than this earliest year, the date is designated as corresponding to the next century. If the year designation is greater than this earliest year, however, the date is designated as corresponding to the current century. The year can then be properly designated as yyyy and the date can be reformatted as yyyymmdd for subsequent data sorting.

7. As evidenced by the computer program dated April 4, 1996 that is attached as Exhibit G, I eventually developed a computer program for date formatting and sorting for dates spanning the turn of the century as claimed in the above referenced patent application. **(See Exhibit G, line 3, designated by an "v")**. Thus, I reduced the claimed invention to practice at least as early as April 4, 1996.

8. I recently became aware of the first and third editions of an IBM article entitled "*The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation.*" The first edition of the article indicates a publication date of October 1995 and the third edition indicates a publication date of May 1996.

9. However, the above statements and information demonstrate that I conceived of a method for date formatting and sorting for dates spanning the turn of the century prior to October 1995. The above statements and attached exhibits also demonstrate that I diligently worked to reduce to practice the claimed method from a date prior to October 1995 to my reduction of practice of the claimed invention at least as early as April 4, 1996.

10. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application of any patent issued thereon.

Bruce Dickens
BRUCE DICKENS

Mar 17, 1998
DATE

Exhibits to Declaration

Exhibit A

```
!   open #1:name 'list.dat', access output
!10 open structure TOOLS  ame "otms_src_dir:TOOLS"
20 extract structure PARTS
        SORT BY PARTS(MODEL)
     end extract
30 print  'PARTS info'
   for each PARTS
        A = PARTS(UNIT_PRICE)*
!       AB = PARTS(UUNIT_PRICE)
!       AA = A[2:7]
!       B = DTYPE(A[2:7])
!       B = VAL(TOOLS(QTY))
!       B = VALID(TOOLS(QTY),NUMBER)
!       IF  B = 1  THEN
     print  PARTS(PARTNO),PARTS(UNIT_PRICE); ' ';PARTS(UUNIT_PRICE)
!       END IF
   next PARTS
   close structure PARTS
!   close #1
50 end
```

D55:[Users.Bruced.Intouch.OTM]OTM.INT

Exhibit B

Directory D55:[USERS.BRUC[illegible].INTOUCH.OTM]

ADD_STR.INT;10	4	28-MAY-1994	10:57:04.38
ADD_STR2.INT;14	2	27-MAY-1994	12:20:27.46
ADD_STR_EX.INT;2	1	16-MAR-1995	13:45:20.70
ADD_STR_EX2.INT;7	2	18-MAR-1995	13:18:50.09
D_STR_EX3.INT;8	2	18-MAR-1995	13:53:38.55
ADD_STR_PRT.INT;6	2	27-MAY-1994	13:49:11.49
ADD_STR_TOL.INT;5	2	27-MAY-1994	13:50:01.88
AR1.INT;4	1	19-OCT-1994	07:24:24.49
AR2.INT;29	2	19-OCT-1994	09:25:50.84
B.COM;3	1	8-JUN-1994	13:23:18.98
CUSTOMER.DAT;2	15	30-MAY-1988	11:44:48.50
CUSTOMER.DEF;1	51	4-MAY-1988	11:34:50.58
CUSTOMER.FDL;3	3	30-JAN-1990	10:27:01.95
CUSTOMER.STR;1	1	4-MAY-1988	11:34:42.57
EXT_PAR_CHL.INT;1	2	19-AUG-1994	14:37:00.12
EXT_PAR_CHLD.INT;1	2	19-AUG-1994	14:20:57.50
EXT_PAR_CLD1.INT;1	2	20-AUG-1994	09:41:20.21
EXT_PRT.INT;1	4	2-FEB-1995	12:48:06.68
GOV.DEF;1	27	23-MAY-1994	13:26:47.33
GOV.FDL;2	4	27-MAY-1994	12:08:01.97
GOV.INT;4	2	25-MAY-1994	13:16:58.17
GOV.STR;1	1	23-MAY-1994	13:25:57.88
GOVMOD.INT;5	39	28-MAY-1994	10:58:36.03
GOVMODSUB.INT;5	44	28-MAY-1994	11:01:44.56
GOVRPT.INT;6	38	28-MAY-1994	11:00:18.57
GOVSPMOD.INT;6	55	28-MAY-1994	11:03:10.78
GOVTPSUB.INT;6	43	28-MAY-1994	11:06:06.87
INVOICE.DAT;2	21	30-MAY-1988	11:46:15.32
INVOICE.DEF;2	51	4-MAY-1988	12:00:53.54
INVOICE.FDL;3	3	30-JAN-1990	10:27:20.33
VOICE.MAIN;2	5	19-OCT-1989	17:04:31.53
VOICE.STR;1	1	4-MAY-1988	12:00:27.50
LEDGER.LIS;1	3	26-MAY-1994	12:06:25.90
LIST.DAT;4	371	27-SEP-1994	07:27:50.38
MAINTAIN.INT;225	313	16-FEB-1993	11:36:00.31
MAINTAIN_DAV.INT;1	315	10-SEP-1992	11:16:26.20
MAINTAIN_HELP.INC;2	19	25-AUG-1989	16:15:31.14
MARGIN.INT;1	1	25-MAY-1994	10:09:38.48
MSAF$RESOURCES.DIR;1	1	31-JAN-1995	15:13:15.62
NANCY1.GUIDE;1	1	16-AUG-1994	14:12:29.86
NANCY1.INT;10	33	18-AUG-1994	13:02:36.17
NONAME.INT;3	1	5-NOV-1994	10:44:54.09
OTM.INT;36	2	12-OCT-1996	09:58:09.60
OTMOUT.INT;2	2	27-SEP-1994	07:27:38.18
PARTE_EXC1.RPT;10	1	27-MAY-1994	12:18:45.99
PARTE_EXC2.RPT;10	1	27-MAY-1994	12:18:46.38
PARTE_EXC3.RPT;10	1	27-MAY-1994	12:18:47.65
PRI_EX.INT;1	1	24-MAY-1994	08:00:07.35
PRI_EX2.INT;12	2	25-MAY-1994	09:10:54.54
PRI_EX3.INT;6	2	25-MAY-1994	09:59:37.23
PTY.CMD;6	1	27-MAY-1994	12:15:58.83
PTY.INT;7	14	27-MAY-1994	11:59:38.69
RR.DAT;1	144	12-AUG-1994	09:54:35.18
RR.DEF;1	27	18-JUL-1994	12:14:17.48
RR.FDL;2	4	19-JUL-1994	06:34:53.54
.MAIN;34	5	13-AUG-1994	14:42:11.00
RR.STR;2	1	18-MAR-1995	13:00:50.58
TOOL1.INT;34	37	20-AUG-1994	13:39:38.83
TOOLS.DAT;2	144	3-JUN-1993	12:20:39.12
TOOLS.DEF;1	27	3-JUN-1993	10:08:30.19
TOOLS.FDL;3	4	2-AUG-1994	10:55:23.28

Exhibit C

```
! Program:    MRF~.int Version 3.0-9
! Package:    In_.ich Utilities
! Author:     BRUCE DICKENS
! Date:       October 1, 1992
! Purpose:    ON-LINE TOOL MATERIAL SYSTEM
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

10      set system : comment "Initializing report"
        ask system : mode mode$
        ask system, symbol 'otms_toolno$':value toolno$
!added one line above   !moved value of global symbol to toolno$
        show_stats = true
        if  mode$ = "BATCH"  or  mode$ = "OTHER"  then  show_stats = false
        string_dtype = 1
        declare dynamic field_data
        max_list = 50
        max_structure = 9
        max_titles% = 20
        dim title$(max_titles%)
        dim real total(max_list,0 to max_list)
        dim real extract_totals(max_list)
        dim break_hold$(0 to max_list)
        dim break_compare$(0 to max_list)
        dim integer page_break(0 to max_list)
        dim integer skip_break_totals(0 to max_list)
        dim integer level_count(0 to max_list)
        dim integer groupcount(0 to max_list)
        dim integer str_found(0 to max_structure)
        dim string break_desc$(0 to max_list)
        dim integer break_pos(0 to max_list)

20      !Special variable declarations go here
        today$ = date$(days(date$),3)
        cntrl_z$ = chr$(26)
        no_current_record = 7305
        out_ch = 2
        st_inx% = 0
        last_break_level% = 1
        no_stats = false
        ask margin old_margin
        ask system, logical 'guide_file_locator' : value guide_file_locator$
        if  guide_file_locator$ <> '' then  &
          open structure locator : name guide_file_locator$
        ! for those site that use a file locator system for placing datafiles

        USR_COUNT = 1

100     gosub main_logic
        end

1000    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! M A I N   L O G I C
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine main_logic

1020    gosub init
        gosub open_files
      do
        do
          gosub extract_init
          if bailout% then exit do
          gosub extract_records
          if bailout% then exit do
          gosub print_init
```

```
      gosub print_report
      if bailout% then exit do
      message "Pages printed: " + str$(page_counter%)
    end do
    close #out_ch
    if bailout% then
      message error: 'Report aborted by user action...'
      delay 3
    else
      gosub display_report
    end if
    if bailout% then exit do
  loop
    close all
    clear

1099    end routine


    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ! I N I T
    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ! start up initialization
    !
    ! Expected:
    !
    ! Result  :
    !
    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    routine init

    margin = 132
    set margin margin
    frame off
    gosub define_constants

    end routine


10000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! D E F I N E   C O N S T A N T S
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine define_constants

10010   ! Special definitions go here
        tsuppress% = false
        making_report = true
        po_print_tag$ = "otm_"
        making_export = false
        making_export_structure = false
        unique_report_name = true
        report_name$ = "OTMS_RPT_DIR:OTM_MRR.LIS"
        report_margin = 255
        building_message$ = "Building report..."
        scr_width% = 132
        levels% = 0
        deepest_level% = 0
        totals_count% = 3
        do_totals% = true
        do_groupcount% = false
        titles% = 3
        lookup_err$ = "."
        spacing% = 1
        cutoff% = 0
```

```
          formsize$ = "66"
          lines_per_page% = 58
          break_hold$(0) = ""
          break_compare$(0) = ""
          page_break(0) = false
          skip_break_totals(0) = false
          break_desc$(0) = ""
          break_pos(0) =  1
          title$(1) = "On-Line Tool Material System"
 !         title$(2) = "SPECIAL TOOLING CARD"
 !         title$(3) =
          title_length% = 37
          report_width% = 126
          narrow% = false

 10099    end routine


 11000    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          ! O P E N   F I L E S
          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          routine open_files

 11020    ! open structures here
          set system : comment "Opening data structures"
          if  guide_file_locator$ <> "" then
            set structure locator, field #1 : key "MRR"
            if  _extracted = 0 then
              message error : "Structure: MRR - Not found in " + guide_file_locatc
r$

              stop
            end if
            z$ = locator(#2)
            if  pos(z$, ":") = 0 then z$ = z$ + ":"
            open structure MRR: name "OTMS_SRC_DIR:MRR", datafile z$ + locator(#1)
          else
            open structure MRR: name "OTMS_SRC_DIR:MRR"
          end if

 11099    end routine



          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          ! E X T R A C T   I N I T
          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          ! initialize variables needed for the extract phase
          !
          ! Expected:
          !
          ! Result  :
          !        rec         = 0
          !        sta         = 0
          !        found       = 0
          !        key_input$  = ''
          !        bailout%      = false
          !
          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          routine extract_init

          rec% = 0
          sta% = 0
          found% = 0
          detail_rec% = 0
```

```
        page_counter% = 0
        key_input$ = ''
        model$ = ''
        bailout% = false


11200   u_prompt$  = "Tool Number? "
        u_default$ = ""
        u_len  = 20
        do
          gosub ask
          if  _exit  then
            bailout% = true
            exit do
          end if
          if  _back  then  repeat do

        tool_subset$ = ucase$(trim$(u_reply$))

        if tool_subset$ <> "" then toolno$ = tool_subset$
        if tool_subset$  = "" then tool_subset$ = toolno$
        set system, symbol 'otms_toolno$':value toolno$
!*added three lines above          !reset toolno$ to u_default$ and
                                   !set global symbol equal to toolno$
                                   !reset u_default$ to toolno$

        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        !  V A L D A T E   W O R K   O R D E R
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

!       set structure MRR, field toolno : partial key tool_subset$
        set structure MRR, field toolno :         key tool_subset$
          if _extracted = 0 then
            clear
             print at 10,35,bold,underline: "Tool subset not on File"
             delay
             repeat do

        end if
        end do

        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


        gosub show_stats
        message "Extracting records..."
        set system : comment "Extracting from MRR"

        end routine


12000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! E X T R A C T   R E C O P D S
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine extract_records

12020   !when exception in
        extract structure MRR
          gosub show_extract_status
          exclude ((cutoff% > 0) and (found% >= cutoff%))
          if  ((sample% > 0) and (found% >= sample%))  then  exit extract
          if bailout% then exit extract
          ! extract lookups go here
          ! LETS go here
          ! INCLUDES go here
```

```
    ! Looks go here
    ! EXTRACTION TOTALS go here

    ADATE$ = MRR(DATE)[5:6] + MRR(DATE)[1:4] *
     include MRR (TOOLNO)[1:len(tool_subset$)] = tool_subset$
    INCLUDE MRR ( TOOLNO )  =  tool_subset$

      when exception in
      sort  by MRR ( DATE )
      sort by ADATE$ **
       use
      end when
      if  _error  then  exclude true
      found% = found% + 1
      end extract

   gosub show_recs_searched
   gosub show_recs_selected
   end routine


   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ! P R I N T   I N I T
   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ! initialize variables needed during the print phase
   !
   ! Expected:
   !
   ! Result  :
   !       The ouput file is opened
   !        start_print_time = time(0)
   !        first_line       = true
   !        key_input$       = ''
   !        page_counter%     = 0
   !
   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  routine print_init

  message building_message$
  set system : comment building_message$

  if  out_ch > 0 and not making_export_structure then
    if  unique_report_name then
      open #out_ch: name report_name$, access output, unique
      ask #out_ch : name report_name$
    else
      open #out_ch: name report_name$, access output
    end if
    set #out_ch: margin report_margin
  end if

  start_print_time = time(0)
  key_input$ = ''
  line_counter%   = lines_per_page%  ! cause first page break
  first_line%   = true

  end routine


  !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  ! P R I N T   R E P O R T:
  !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  routine print_report
```

```
        LET HOURS = 0
        TOTAL(1,LEVEL%) = 0
        TOTAL(2,LEVEL%) = 0
        TOTAL(3,LEVEL%) = 0
13020   for each MRR
          gosub show_report_status
          if  bailout% = true  then exit for
          ! Do file lookups

        print toolno, hours, mrr ( fab_cost )

        HOURS = HOURS + MRR ( FAB_COST )

          gosub set_user_defined
          gosub add_totals
          gosub print_detail_line
          first_line% = false
          level_count(0) = level_count(0) + 1
        next MRR
        level% = 0
        gosub print_totals
        if  out_ch > 0  then close #out_ch

13099   end routine


13500   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! R E L A T E   M A N Y
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine relate_many


        end routine


14000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! D I S P L A Y   R E P O R T
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        !       u_dispatch$ = routine to dispatch to after clear screen
        !                     (u_dispatch$ is cleared on reference)
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine display_report

14020   if  not making_report  then
          gosub show_stats
          delay
          exit routine
        end if
        if  out_ch = 0  then exit routine
        set system : comment "Displaying report"
        u_str$ = report_name$
        u_scr_width% = margin
        u_dispatch$ = 'show_stats'
        gosub prnt_ask_option

14099   end routine


21000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! N E W   P A G E
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine new_page
```

```
21020   if  not making_report  then  exit routine
        page_counter% = page_counter% + 1
        gosub show_recs_printed
        gosub show_pages_printed
        gosub show_eoj
        print #out_ch: chr$(12);

        gosub check_title_substitutes

        page$ = "Page " + str$(page_counter%)
        p% = len(page$)
        t% = len(title$(1))
        rpt_width% = max(report_width%,t%+22%)
        rpt_width% = max(rpt_width%, len(title$(2)))
        rpt_width% = max(rpt_width%, len(title$(3)))

        z% = (rpt_width% - t%) / 2
        if  z% < 13%  then  z% = 13% ! leave room for the date
        print #out_ch: today$;
          print #out_ch: tab(z% + 1 ); title$(1);
          print #out_ch: tab(rpt_width% - p% + 1); page$; chr$(0)
          line_counter% = 1

       print #out_ch:tab(1); "SPECIAL TOOLING CARD";
       print #out_ch:tab(114); "COST HISTORY CARD"
       for i% = 2 to titles%
         z% = (rpt_width% - len(title$(i%)))/2
         if  z% < 1%  then  z% = 1%
         print #out_ch: tab(z% + 1); title$(i%); chr$(0)
         line_counter% = line_counter% + 1
       next i%

       print #out_ch:tab(2); "TOOL NBR:";  TAB(13); MRR(TOOLNO)

       print #out_ch: chr$(0)
       line_counter% = line_counter% + 1

       print #out_ch : tab(1); "DATE";
       print #out_ch : tab(10); "TWO ";
       print #out_ch : tab(21); "CCN";
       print #out_ch : tab(32); "D E S C R I P T I O N";
       print #out_ch : tab(88); "MAT_COST";
       print #out_ch : tab(103); "FAB_COST";
       print #out_ch : tab(117); "TOTAL";
       print #out_ch :  chr$(0)
       line_counter% = line_counter% + 1
       print #out_ch : tab(1);  repeat$("-", 8);
       print #out_ch : tab(10); repeat$("-", 10);
       print #out_ch : tab(21); repeat$("-", 10);
       print #out_ch : tab(32); repeat$("-", 50);
       print #out_ch : tab(88); repeat$("-", 12);
       print #out_ch : tab(103); repeat$("-", 12);
       print #out_ch : tab(117); repeat$("-", 12);
       print #out_ch :  chr$(0)
       line_counter% = line_counter% + 1
       first_page_line% = true
21099  end routine


21100  !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       ! C H E C K   T I T L E   S U B S T I T U T E S
       !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       routine check_title_substitutes
```

```
21120   when exception in
        use
          continue
        end when

21199   end routine

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! S E T   U S E R   D E F I N E D %%%%%%%%%%%%%%%%%%%%%%%%%%%%
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine set_user_defined

22020   ! User-defined variables get calculated here
        when exception in
        use
          if  extype = no_current_record  then continue
        end when
22099   end routine
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
24000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! A D D   T O T A L S %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine add_totals

24020   when exception in
          for level% = 0 to levels%
            ! Add level totals here
            total(1,level%) = total(1,level%) + MRR(MAT_COST)
            total(2,level%) = total(2,level%) + MRR(FAB_COST)
            total(3,level%) = total(3,level%) + MRR(TOTAL)
          next level%
        use
          if  extype = no_current_record  then continue
        end when
24099   end routine

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! P R I N T   D E T A I L   L I N E %%%%%%%%%%%%%%%%%%%%%%%%%%%
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! Print detail line: col_data$(col) for each column %%%%%%%%%%%
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine print_detail_line

26020   ! Print detail lines here
        if  line_counter% >= lines_per_page%  then gosub new_page
        data_printed% = false
        when exception in
        col_len% = 8
          print #out_ch : tab(1);
          print #out_ch, using "%%~~%%~~%%" : MRR ( DATE );
          data_printed% = true

        col_len% = 10
          print #out_ch : tab(10);
          print #out_ch, using "<#########" : MRR ( TWO );
          data_printed% = true

        col_len% = 10
          print #out_ch : tab(21);
          print #out_ch, using "<#########" : MRR ( CCN );
          data_printed% = true
```

```
        col_len% = 50
          print #out_ch :  :ab(32);
          print #out_ch, using "<####################################################
##" : MRR ( CHG_DESC );
          data_printed% = true

        col_len% = 12
          print #out_ch : tab(88);
          print #out_ch, using "$##,###,###" : MRR ( MAT_COST );
          data_printed% = true

        col_len% = 12
          print #out_ch : tab(102);
          print #out_ch : "HRS" ; MRR ( FAB_COST );
          data_printed% = true

        col_len% = 12
          print #out_ch : tab(117);
          print #out_ch, using "$##,###,###" : MRR ( TOTAL );
          data_printed% = true

          print #out_ch :
          line_counter% = line_counter% + 1

        use
          print #out_ch: repeat$(lookup_err$, col_len%);
          continue
        end when

        for z% = 1 to spacing%
          print #out_ch:
          line_counter% = line_counter% + 1
        next z%

        data_printed% = false
        first_page_line% = false
        last_break_level% = 0

26099   end routine

27000   !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! P R I N T   T O T A L S
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        routine print_totals

27020   if  tsuppress% and (level_count(level%) = 1) and (level% = levels%) &
            then exit routine
        if  skip_break_totals(level%) then exit routine
        if  line_counter% >= lines_per_page%  then gosub new_page

        print #out_ch : tab(88); repeat$("=", 12);
        print #out_ch : tab(103); repeat$("=", 12);
        print #out_ch : tab(117); repeat$("=", 12);
        print #out_ch :
        line_counter% = line_counter% + 1

        if  break_pos(level%) <> 0 then &
          print #out_ch : tab(break_pos(level%)); break_desc$(level%);
        when exception in
          col_len% = 12
          print #out_ch : tab(87);
          print #out_ch, using "$###,###,###" : TOTAL(1,LEVEL%);

          col_len% = 12
          print #out_ch : tab(102);
          print #out_ch : "HRS" ; tab(106) ; HOURS;
```

```
            col_len% = 12
            print #out_ch, tab(116);
            print #out_ch, using "$###,###,###" : TOTAL(3,LEVEL%);

          use
            print #out_ch: repeat$(lookup_err$, col_len%);
            continue
          end when

          print #out_ch:
          line_counter% = line_counter% + 1

 27099    end routine


 81600    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          ! A S K
          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          !
          ! Ask the expected prompt.
          !
          ! EXPECTED:
          !       u_prompt$ = question being asked
          !       u_default$ = the default
          !       u_len = the length.  Doesn't reset any of these.
          !       u_help_key$ = the help key (optional; will use the prompt
          !          with spaces changed to _ and "?:" removed)
          !       u_required = true if a response is necessary
          !
          ! RESULT:
          !       u_reply$  = user's reply
          !       u_help+key$ is reset to ""
          !       u_required = true
          !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   ask:

 81620    clear area 21,1,21,old_margin
          if  u_help_key$ = ""  then &
            u_help_key$ = change$(trim$(change$(u_prompt$,'?:')),' ","_")
          do

          if u_default$ <> "" then toolno$ = u_default$
          if u_default$ = "" then u_default$ = toolno$
          set system, symbol 'otms_toolno$':value toolno$
 !*added three lines above        !reset toolno$ to u_default$ and
                                  !set global symbol equal to toolno$
                                  !reset u_default$ to toolno$


            u_reply$ = ""
            line input at 21,1, &
                       prompt u_prompt$, &
                       default u_default$, &
                       length u_len: u_reply$
            if  _help  then
              h_key$ = u_help_key$
 !             gosub h_help
              if  u_len = 0  then clear area 21,1,21,old_margin
              repeat do
            end if
            if  _exit or _back  then exit do
            if  u_reply$ = "" and u_required  then  repeat do
            clear area 21,1,21,old_margin

          end do
          u_help_key$ = ""
```

```
81699   return
83000   go to 11200
90000   %include "usr_root:[software.intouch]guide_shell_int1.inc"
99999   end
```

Exhibit D

Directory D4:[PRODUCTION.OTMS.SRC]

B.COM;2	1	8-JUN-1994	13:26:20.35
BARCODE.DIR;1	1	6-APR-1996	09:30:26.76
CENT.INT;135	6	4-MAY-1996	13:32:37.86
CONTRACT_NO.INT;8	5	6-APR-1996	09:52:46.17
CONVERT1.INT;2	2	5-OCT-1996	12:39:48.66
CONVERT2.INT;2	2	25-OCT-1996	08:23:26.72
CONVERT4.INT;1	5	17-OCT-1996	12:03:09.65
CONVERT5.INT;3	5	25-OCT-1996	14:40:23.63
FIXMRR.INT;2	42	6-FEB-1995	14:30:03.82
GOV.DEF;1	27	8-JUN-1994	07:37:56.16
GOV.STR;1	1	8-JUN-1994	07:28:37.26
GOVMOD.INT;7	39	15-JUL-1995	13:02:36.61
GOVMODSUB.INT;7	44	30-MAY-1995	12:27:59.16
GOVRPT.INT;8	38	15-JUL-1995	13:03:14.65
GOVSPMOD.INT;8	55	30-MAY-1995	12:28:51.47
GOVTPSUB.INT;8	43	30-MAY-1995	12:29:43.84
INAMOD.INT;2	38	20-OCT-1995	10:19:43.40
INASPMOD.INT;2	56	3-NOV-1995	13:47:33.33
INATOL.INT;2	39	20-OCT-1995	10:20:38.18
INQMRR_MAIN.INT;2	316	18-SEP-1995	12:30:39.88
INQOTMS000.INT;4	12	19-SEP-1995	07:55:37.40
INQPART_MAIN.INT;2	315	18-SEP-1995	12:34:55.08
INQRR_MAIN.INT;3	316	18-SEP-1995	12:38:22.09
INQTOOL_MAIN.INT;1	318	18-SEP-1995	12:04:50.50
INQTXT_MAIN.INT;2	316	18-SEP-1995	12:42:11.01
LAST_INV.DAT;6	539	4-MAY-1996	13:21:42.93
MAINTAIN.INT;1	315	10-SEP-1992	11:16:26.20
MODELRPT.INT;10	35	3-JUN-1993	09:25:56.49
MODSUB1.INT;40	40	3-JUN-1993	12:48:05.47
MOD_MASS.DAT;5	0	6-APR-1996	09:52:52.46
MRR.DEF;1	27	16-JUL-1994	13:04:00.40
MRR.INT;80	42	15-JUL-1995	12:57:52.69
MRR.MAIN;33	14	2-SEP-1994	12:30:38.82
MRR.STR;3	1	20-SEP-1994	13:21:53.77
MRR_MAIN.INT;7	316	20-JUL-1995	14:37:44.68
O.LIS;4	7	25-OCT-1996	15:25:58.51
OTMS000.INT;31	13	25-OCT-1996	10:57:15.59
OTMS_SRC_DIR.STR;1	1	21-OCT-1995	11:36:39.85
OUT.LIS;8	1381	6-APR-1996	09:21:24.28
PART2.DEF;1	27	16-OCT-1996	10:09:41.12
PART2.FDL;2	4	17-OCT-1996	07:52:33.67
PART2.MAIN;8	13	25-OCT-1996	10:21:33.98
PART2.STR;1	1	16-OCT-1996	09:32:41.15
PARTMOD.INT;3	35	15-JUL-1995	13:01:23.06
PARTMOD2.INT;18	36	9-NOV-1996	09:34:08.32
PARTMODSUB.INT;4	40	9-NOV-1996	10:08:49.12
PARTMODSUB2.INT;9	40	9-NOV-1996	10:11:21.45
PARTRPT.INT;11	35	9-NOV-1996	10:02:18.37
PARTRPT.INT_IMG;1	854	9-NOV-1996	10:03:09.35
PARTRPT2.INT;8	35	9-NOV-1996	10:05:29.92
PARTS.DAT;1	144	7-JUN-1994	15:49:13.99
PARTS.DEF;2	27	7-JUN-1994	15:41:34.28
PARTS.FDL;2	4	5-OCT-1996	11:44:30.77
PARTS.MAIN;2	13	5-OCT-1996	12:32:23.86
PARTS.STR;1	1	7-JUN-1994	15:42:48.74
PARTSPMOD.INT;5	51	12-OCT-1996	10:45:04.97
PARTSPMOD.INT_IMG;1	854	25-OCT-1996	10:52:31.13
PARTSPMOD2.INT;49	52	9-NOV-1996	10:53:21.70
PARTSUB.INT;3	40	30-MAY-1995	12:25:44.86
PARTSUB2.INT;12	40	9-NOV-1996	11:00:17.56
PART_MAIN.INT;4	315	17-OCT-1996	13:13:23.64
REPORT1.GUIDE;1	1	25-SEP-1992	12:20:14.38

Exhibit E

```
1       !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! Program:       Shop Day Generator Utility
        ! Package:       Macpac/D Control File
        ! Author :       Bruce Dickens
        ! Date   :       March 13, 1995
        !Purpose :       Provide a method to generate 9998
        !                contol file records for any year.
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


1000    !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        ! M A I N   L O G I C   S E C T I O N
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        !
        ! Ask for any calendar year and generate all MacPac/D
        ! 9998 control file records for that year for input into
        ! the MacPac\D update facility.
        !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        input 'Calendar Year to Generate? (YYYY)': Year%          !BDickens000001
        open #1:name 'calgen.lis', access output                  !BDickens000002
                        option base 0                              !BDickens000003
                        dim wkdyr$(28)                             !BDickens000004
                        wkdyr$(0)  = "fri"                         !BDickens000005
                        wkdyr$(1)  = "sun"                         !BDickens000006
                        wkdyr$(2)  = "mon"                         !BDickens000007
                        wkdyr$(3)  = "tue"                         !BDickens000008
                        wkdyr$(4)  = "wed"                         !BDickens000009
                        wkdyr$(5)  = "fri"                         !BDickens000010
                        wkdyr$(6)  = "sat"                         !BDickens000011
                        wkdyr$(7)  = "sun"                         !BDickens000012
                        wkdyr$(8)  = "mon"                         !BDickens000013
                        wkdyr$(9)  = "wed"                         !BDickens000014
                        wkdyr$(10)  = "thu"                        !BDickens000015
                        wkdyr$(11)  = "fri"                        !BDickens000016
                        wkdyr$(12)  = "sat"                        !BDickens000017
                        wkdyr$(13)  = "mon"                        !BDickens000018
                        wkdyr$(14)  = "tue"                        !BDickens000019
                        wkdyr$(15)  = "wed"                        !BDickens000020
                        wkdyr$(16)  = "thu"                        !BDickens000021
                        wkdyr$(17)  = "sat"                        !BDickens000022
                        wkdyr$(18)  = "sun"                        !BDickens000023
                        wkdyr$(19)  = "mon"                        !BDickens000024
                        wkdyr$(20)  = "tue"                        !BDickens000025
                        wkdyr$(21)  = "thu"                        !BDickens000026
                        wkdyr$(22)  = "fri"                        !BDickens000027
                        wkdyr$(23)  = "sat"                        !BDickens000028
                        wkdyr$(24)  = "sun"                        !BDickens000029
                        wkdyr$(25)  = "tue"                        !BDickens000030
                        wkdyr$(26)  = "wed"                        !BDickens000031
                        wkdyr$(27)  = "thu"                        !BDickens000032
                        wkdyr$(28)  = "fri"                        !BDickens000033
        a = year%/28                                               !BDickens000034
        c = mod(year%,28)                                          !BDickens000035
          b = year%/4                                              !BDickens000036
          d = mod (year%,4)                                        !BDickens000037
          if d = 0 then                                            !BDickens000038
            e = 12                                                 !BDickens000039
          else                                                     !BDickens000040
            e = 11                                                 !BDickens000041
          end if                                                   !BDickens000042

        days = 0                                                   !BDickens000043
        f = c                                                      !BDickens000044

5000            option base 0                                      !BDickens000045
```

```
        dim wkd$(7)                                   !BDickens000046
            wkd$(0)  = "fri"                          !BDickens000047
            wkd$(1)  = "sat"                          !BDickens000048
            wkd$(2)  = "sun"                          !BDickens000049
            wkd$(3)  = "mon"                          !BDickens000050
            wkd$(4)  = "tue"                          !BDickens000051
            wkd$(5)  = "wed"                          !BDickens000052
            wkd$(6)  = "thu"                          !BDickens000053
    for g = 0 to 6                                    !BDickens000054
      if wkd$(g)  = wkdyr$(f) then i = g              !BDickens000055
    next g                                            !BDickens000056

6000    dim mo$(12)                                   !BDickens000057
            mo$(1)  =   "JAN"                         !BDickens000058
            mo$(2)  =   "FEB"                         !BDickens000058
            mo$(3)  =   "MAR"                         !BDickens000059
            mo$(4)  =   "APR"                         !BDickens000060
            mo$(5)  =   "MAY"                         !BDickens000061
            mo$(6)  =   "JUN"                         !BDickens000062
            mo$(7)  =   "JUL"                         !BDickens000063
            mo$(8)  =   "AUG"                         !BDickens000064
            mo$(9)  =   "SEP"                         !BDickens000065
            mo$(10)  =  "OCT"                         !BDickens000066
            mo$(11)  =  "NOV"                         !BDickens000067
            mo$(12)  =  "DEC"                         !BDickens000068


    day% = 0                                          !BDickens000069
    mo% = 0                                           !BDickens000070
    shopday% = 0                                      !BDickens000071
    count% = 0                                        !BDickens000072


    do until mo% = 12                                 !BDickens000073
      mo% = mo% + 1                                   !BDickens000074
        for day% = 1 to 31                            !BDickens000075
          h = mod(i,7)                                !BDickens000076
          cal$ = str$(year%)  + lpad$(str$(mo%),2,'0') &   !BDickens000077
                              + lpad$(str$(day%),2,'0')    !BDickens000078


12000 !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      ! Holiday Section
      !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      ! This section of code assigns shop days and lists
      ! calendar days.
      !%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if d = 0 then
           goto 12100
        else
          if cal$ = str$(year%)  + "0229"   then goto 13500
        end if

12100     if cal$ = str$(year%)  + "0230"   then goto 13500
          if cal$ = str$(year%)  + "0231"   then goto 13500
          if cal$ = str$(year%)  + "0431"   then goto 13500
          if cal$ = str$(year%)  + "0631"   then goto 13500
          if cal$ = str$(year%)  + "0931"   then goto 13500
          if cal$ = str$(year%)  + "1131"   then goto 13500
          if cal$ = str$(year%)


          if wkd$(h)  = "sun" and cal$ = str$(year%)  + "0101" then
              shopday% = 1
              goto 13000
```

```
            if wkd$(h) = "mon" and cal$ = str$(year%) + "0907" then goto 13000

!               Thanksgiving (4th Thursday in November)
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1122" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1123" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1123" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1124" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1124" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1125" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1125" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1126" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1126" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1127" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1127" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1128" then goto 13000
            if wkd$(h) = "thu" and cal$ = str$(year%) + "1128" then goto 13000
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1129" then goto 13000

!               Christmas Break (December 25th)
            if wkd$(h) = "fri" and cal$ = str$(year%) + "1222" then goto 13000
            if wkd$(h) = "mon" and cal$ = str$(year%) + "1223" then goto 13000
            if cal$ = str$(year%) + "1224"  then goto 13000
            if cal$ = str$(year%) + "1225"  then goto 13000
            if cal$ = str$(year%) + "1226"  then goto 13000
            if cal$ = str$(year%) + "1227"  then goto 13000
            if cal$ = str$(year%) + "1228"  then goto 13000
            if cal$ = str$(year%) + "1229"  then goto 13000
            if cal$ = str$(year%) + "1230"  then goto 13000
            if cal$ = str$(year%) + "1231"  then goto 13000

12500         shopday% = shopday% + 1

13000       i = i + 1
              count% = count% + 1
                k$ = cal$[3:4] + lpad$(str$(shopday%),3,'0')
                m$ = mo$(mo%) + ' ' + cal$[7:8] + ', ' + cal$[1:4]
                print #1:  '99982'; cal$;'      '; cal$[3:8]; k$ ;  &
                                  ';m$;'                              '; 'C'

                                                                    !BDickens000079
13500     next day%                                                 !BDickens000080
        loop
                                                                    !BDickens000081
99999   end
```

Exhibit F

Attachment 1

Menu request:

```
BRUCED> sint calgen
Calendar Year to Generate? (YYYY)? 1996
```

1. CALGEN is the executable.
2. Calendar Year to Generate? (YYYY)? is the prompt.
3. 1996 is the reply.

The following pages are the formatted output for input to the MacPac control file and calendar update programs. I would prefer to have my algorithm put in place of all those pages of 9998 records.

The program is designed very simply. The intouch code is less than four pages long. The majority of the code is devoted to holiday definitions.

9998219960101	96010196001	JAN 01, 1996	C
9998219960102	96010296001	JAN 02, 1996	C
9998219960103	96010396002	JAN 03, 1996	C
9998219960104	96010496003	JAN 04, 1996	C
9998219960105	96010596004	JAN 05, 1996	C
9998219960106	96010696004	JAN 06, 1996	C
9998219960107	96010796004	JAN 07, 1996	C
9998219960108	96010896005	JAN 08, 1996	C
9998219960109	96010996006	JAN 09, 1996	C
9998219960110	96011096007	JAN 10, 1996	C
9998219960111	96011196008	JAN 11, 1996	C
9998219960112	96011296009	JAN 12, 1996	C
9998219960113	96011396009	JAN 13, 1996	C
9998219960114	96011496009	JAN 14, 1996	C
9998219960115	96011596010	JAN 15, 1996	C
9998219960116	96011696011	JAN 16, 1996	C
9998219960117	96011796012	JAN 17, 1996	C
9998219960118	96011896013	JAN 18, 1996	C
9998219960119	96011996014	JAN 19, 1996	C
9998219960120	96012096014	JAN 20, 1996	C
9998219960121	96012196014	JAN 21, 1996	C
9998219960122	96012296015	JAN 22, 1996	C
9998219960123	96012396016	JAN 23, 1996	C
9998219960124	96012496017	JAN 24, 1996	C
9998219960125	96012596018	JAN 25, 1996	C
9998219960126	96012696019	JAN 26, 1996	C
9998219960127	96012796019	JAN 27, 1996	C
9998219960128	96012896019	JAN 28, 1996	C
9998219960129	96012996020	JAN 29, 1996	C
9998219960130	96013096021	JAN 30, 1996	C
9998219960131	96013196022	JAN 31, 1996	C
9998219960201	96020196023	FEB 01, 1996	C
9998219960202	96020296024	FEB 02, 1996	C
9998219960203	96020396024	FEB 03, 1996	C
9998219960204	96020496024	FEB 04, 1996	C
9998219960205	96020596025	FEB 05, 1996	C
9998219960206	96020696026	FEB 06, 1996	C
9998219960207	96020796027	FEB 07, 1996	C
9998219960208	96020896028	FEB 08, 1996	C
9998219960209	96020996029	FEB 09, 1996	C
9998219960210	96021096029	FEB 10, 1996	C
9998219960211	96021196029	FEB 11, 1996	C
9998219960212	96021296030	FEB 12, 1996	C
9998219960213	96021396031	FEB 13, 1996	C
9998219960214	96021496032	FEB 14, 1996	C
9998219960215	96021596033	FEB 15, 1996	C
9998219960216	96021696034	FEB 16, 1996	C
9998219960217	96021796034	FEB 17, 1996	C
9998219960218	96021896034	FEB 18, 1996	C
9998219960219	96021996035	FEB 19, 1996	C
9998219960220	96022096036	FEB 20, 1996	C
9998219960221	96022196037	FEB 21, 1996	C
9998219960222	96022296038	FEB 22, 1996	C
9998219960223	96022396039	FEB 23, 1996	C
9998219960224	96022496039	FEB 24, 1996	C
9998219960225	96022596039	FEB 25, 1996	C
9998219960226	96022696040	FEB 26, 1996	C
9998219960227	96022796041	FEB 27, 1996	C
9998219960228	96022896042	FEB 28, 1996	C
9998219960229	96022996043	FEB 29, 1996	C
9998219960301	96030196044	MAR 01, 1996	C
9998219960302	96030296044	MAR 02, 1996	C
9998219960303	96030396044	MAR 03, 1996	C
9998219960304	96030496045	MAR 04, 1996	C
9998219960305	96030596046	MAR 05, 1996	C
9998219960306	96030696047	MAR 06, 1996	C

9998219960307	96030796048	MAR 07, 1996	C
9998219960308	96030896049	MAR 08, 1996	C
9998219960309	96030996049	MAR 09, 1996	C
9998219960310	96031096049	MAR 10, 1996	C
9998219960311	96031196050	MAR 11, 1996	C
9998219960312	96031296051	MAR 12, 1996	C
9998219960313	96031396052	MAR 13, 1996	C
9998219960314	96031496053	MAR 14, 1996	C
9998219960315	96031596054	MAR 15, 1996	C
9998219960316	96031696054	MAR 16, 1996	C
9998219960317	96031796054	MAR 17, 1996	C
9998219960318	96031896055	MAR 18, 1996	C
9998219960319	96031996056	MAR 19, 1996	C
9998219960320	96032096057	MAR 20, 1996	C
9998219960321	96032196058	MAR 21, 1996	C
9998219960322	96032296059	MAR 22, 1996	C
9998219960323	96032396059	MAR 23, 1996	C
9998219960324	96032496059	MAR 24, 1996	C
9998219960325	96032596060	MAR 25, 1996	C
9998219960326	96032696061	MAR 26, 1996	C
9998219960327	96032796062	MAR 27, 1996	C
9998219960328	96032896063	MAR 28, 1996	C
9998219960329	96032996064	MAR 29, 1996	C
9998219960330	96033096064	MAR 30, 1996	C
9998219960331	96033196064	MAR 31, 1996	C
9998219960401	96040196065	APR 01, 1996	C
9998219960402	96040296066	APR 02, 1996	C
9998219960403	96040396067	APR 03, 1996	C
9998219960404	96040496068	APR 04, 1996	C
9998219960405	96040596069	APR 05, 1996	C
9998219960406	96040696069	APR 06, 1996	C
9998219960407	96040796069	APR 07, 1996	C
9998219960408	96040896070	APR 08, 1996	C
9998219960409	96040996071	APR 09, 1996	C
9998219960410	96041096072	APR 10, 1996	C
9998219960411	96041196073	APR 11, 1996	C
9998219960412	96041296074	APR 12, 1996	C
9998219960413	96041396074	APR 13, 1996	C
9998219960414	96041496074	APR 14, 1996	C
9998219960415	96041596075	APR 15, 1996	C
9998219960416	96041696076	APR 16, 1996	C
9998219960417	96041796077	APR 17, 1996	C
9998219960418	96041896078	APR 18, 1996	C
9998219960419	96041996079	APR 19, 1996	C
9998219960420	96042096079	APR 20, 1996	C
9998219960421	96042196079	APR 21, 1996	C
9998219960422	96042296080	APR 22, 1996	C
9998219960423	96042396081	APR 23, 1996	C
9998219960424	96042496082	APR 24, 1996	C
9998219960425	96042596083	APR 25, 1996	C
9998219960426	96042696084	APR 26, 1996	C
9998219960427	96042796084	APR 27, 1996	C
9998219960428	96042896084	APR 28, 1996	C
9998219960429	96042996085	APR 29, 1996	C
9998219960430	96043096086	APR 30, 1996	C
9998219960501	96050196087	MAY 01, 1996	C
9998219960502	96050296088	MAY 02, 1996	C
9998219960503	96050396089	MAY 03, 1996	C
9998219960504	96050496089	MAY 04, 1996	C
9998219960505	96050596089	MAY 05, 1996	C
9998219960506	96050696090	MAY 06, 1996	C
9998219960507	96050796091	MAY 07, 1996	C
9998219960508	96050896092	MAY 08, 1996	C
9998219960509	96050996093	MAY 09, 1996	C
9998219960510	96051096094	MAY 10, 1996	C
9998219960511	96051196094	MAY 11, 1996	C

9998219960512	96051296094	MAY 12, 1996	C
9998219960513	96051396095	MAY 13, 1996	C
9998219960514	96051496096	MAY 14, 1996	C
9998219960515	96051596097	MAY 15, 1996	C
9998219960516	96051696098	MAY 16, 1996	C
9998219960517	96051796099	MAY 17, 1996	C
9998219960518	96051896099	MAY 18, 1996	C
9998219960519	96051996099	MAY 19, 1996	C
9998219960520	96052096100	MAY 20, 1996	C
9998219960521	96052196101	MAY 21, 1996	C
9998219960522	96052296102	MAY 22, 1996	C
9998219960523	96052396103	MAY 23, 1996	C
9998219960524	96052496104	MAY 24, 1996	C
9998219960525	96052596104	MAY 25, 1996	C
9998219960526	96052696104	MAY 26, 1996	C
9998219960527	96052796104	MAY 27, 1996	C
9998219960528	96052896105	MAY 28, 1996	C
9998219960529	96052996106	MAY 29, 1996	C
9998219960530	96053096107	MAY 30, 1996	C
9998219960531	96053196108	MAY 31, 1996	C
9998219960601	96060196108	JUN 01, 1996	C
9998219960602	96060296108	JUN 02, 1996	C
9998219960603	96060396109	JUN 03, 1996	C
9998219960604	96060496110	JUN 04, 1996	C
9998219960605	96060596111	JUN 05, 1996	C
9998219960606	96060696112	JUN 06, 1996	C
9998219960607	96060796113	JUN 07, 1996	C
9998219960608	96060896113	JUN 08, 1996	C
9998219960609	96060996113	JUN 09, 1996	C
9998219960610	96061096114	JUN 10, 1996	C
9998219960611	96061196115	JUN 11, 1996	C
9998219960612	96061296116	JUN 12, 1996	C
9998219960613	96061396117	JUN 13, 1996	C
9998219960614	96061496118	JUN 14, 1996	C
9998219960615	96061596118	JUN 15, 1996	C
9998219960616	96061696118	JUN 16, 1996	C
9998219960617	96061796119	JUN 17, 1996	C
9998219960618	96061896120	JUN 18, 1996	C
9998219960619	96061996121	JUN 19, 1996	C
9998219960620	96062096122	JUN 20, 1996	C
9998219960621	96062196123	JUN 21, 1996	C
9998219960622	96062296123	JUN 22, 1996	C
9998219960623	96062396123	JUN 23, 1996	C
9998219960624	96062496124	JUN 24, 1996	C
9998219960625	96062596125	JUN 25, 1996	C
9998219960626	96062696126	JUN 26, 1996	C
9998219960627	96062796127	JUN 27, 1996	C
9998219960628	96062896128	JUN 28, 1996	C
9998219960629	96062996128	JUN 29, 1996	C
9998219960630	96063096128	JUN 30, 1996	C
9998219960701	96070196129	JUL 01, 1996	C
9998219960702	96070296130	JUL 02, 1996	C
9998219960703	96070396131	JUL 03, 1996	C
9998219960704	96070496131	JUL 04, 1996	C
9998219960705	96070596132	JUL 05, 1996	C
9998219960706	96070696132	JUL 06, 1996	C
9998219960707	96070796132	JUL 07, 1996	C
9998219960708	96070896133	JUL 08, 1996	C
9998219960709	96070996134	JUL 09, 1996	C
9998219960710	96071096135	JUL 10, 1996	C
9998219960711	96071196136	JUL 11, 1996	C
9998219960712	96071296137	JUL 12, 1996	C
9998219960713	96071396137	JUL 13, 1996	C
9998219960714	96071496137	JUL 14, 1996	C
9998219960715	96071596138	JUL 15, 1996	C
9998219960716	96071696139	JUL 16, 1996	C

9998219960921	96092196186	SEP 21, 1996	C
9998219960922	96092296186	SEP 22, 1996	C
9998219960923	96092396187	SEP 23, 1996	C
9998219960924	96092496188	SEP 24, 1996	C
9998219960925	96092596189	SEP 25, 1996	C
9998219960926	96092696190	SEP 26, 1996	C
9998219960927	96092796191	SEP 27, 1996	C
9998219960928	96092896191	SEP 28, 1996	C
9998219960929	96092996191	SEP 29, 1996	C
9998219960930	96093096192	SEP 30, 1996	C
9998219961001	96100196193	OCT 01, 1996	C
9998219961002	96100296194	OCT 02, 1996	C
9998219961003	96100396195	OCT 03, 1996	C
9998219961004	96100496196	OCT 04, 1996	C
9998219961005	96100596196	OCT 05, 1996	C
9998219961006	96100696196	OCT 06, 1996	C
9998219961007	96100796197	OCT 07, 1996	C
9998219961008	96100896198	OCT 08, 1996	C
9998219961009	96100996199	OCT 09, 1996	C
9998219961010	96101096200	OCT 10, 1996	C
9998219961011	96101196201	OCT 11, 1996	C
9998219961012	96101296201	OCT 12, 1996	C
9998219961013	96101396201	OCT 13, 1996	C
9998219961014	96101496202	OCT 14, 1996	C
9998219961015	96101596203	OCT 15, 1996	C
9998219961016	96101696204	OCT 16, 1996	C
9998219961017	96101796205	OCT 17, 1996	C
9998219961018	96101896206	OCT 18, 1996	C
9998219961019	96101996206	OCT 19, 1996	C
9998219961020	96102096206	OCT 20, 1996	C
9998219961021	96102196207	OCT 21, 1996	C
9998219961022	96102296208	OCT 22, 1996	C
9998219961023	96102396209	OCT 23, 1996	C
9998219961024	96102496210	OCT 24, 1996	C
9998219961025	96102596211	OCT 25, 1996	C
9998219961026	96102696211	OCT 26, 1996	C
9998219961027	96102796211	OCT 27, 1996	C
9998219961028	96102896212	OCT 28, 1996	C
9998219961029	96102996213	OCT 29, 1996	C
9998219961030	96103096214	OCT 30, 1996	C
9998219961031	96103196215	OCT 31, 1996	C
9998219961101	96110196216	NOV 01, 1996	C
9998219961102	96110296216	NOV 02, 1996	C
9998219961103	9611039621[illegible]	NOV 03, 1996	C
9998219961104	96110496217	NOV 04, 1996	C
9998219961105	96110596218	NOV 05, 1996	C
9998219961106	96110696219	NOV 06, 1996	C
9998219961107	96110796220	NOV 07, 1996	C
9998219961108	96110896221	NOV 08, 1996	C
9998219961109	96110996221	NOV 09, 1996	C
9998219961110	96111096221	NOV 10, 1996	C
9998219961111	96111196222	NOV 11, 1996	C
9998219961112	96111296223	NOV 12, 1996	C
9998219961113	96111396224	NOV 13, 1996	C
9998219961114	96111496225	NOV 14, 1996	C
9998219961115	96111596226	NOV 15, 1996	C
9998219961116	96111696226	NOV 16, 1996	C
9998219961117	96111796226	NOV 17, 1996	C
9998219961118	96111896227	NOV 18, 1996	C
9998219961119	96111996228	NOV 19, 1996	C
9998219961120	96112096229	NOV 20, 1996	C
9998219961121	96112196230	NOV 21, 1996	C
9998219961122	96112296231	NOV 22, 1996	C
9998219961123	96112396231	NOV 23, 1996	C
9998219961124	96112496231	NOV 24, 1996	C
9998219961125	96112596232	NOV 25, 1996	C

9998219961126	96112696233	NOV 26, 1996	C
9998219961127	96112796234	NOV 27, 1996	C
9998219961128	96112896234	NOV 28, 1996	C
9998219961129	96112996234	NOV 29, 1996	C
9998219961130	96113096234	NOV 30, 1996	C
9998219961201	96120196234	DEC 01, 1996	C
9998219961202	96120296235	DEC 02, 1996	C
9998219961203	96120396236	DEC 03, 1996	C
9998219961204	96120496237	DEC 04, 1996	C
9998219961205	96120596238	DEC 05, 1996	C
9998219961206	96120696239	DEC 06, 1996	C
9998219961207	96120796239	DEC 07, 1996	C
9998219961208	96120896239	DEC 08, 1996	C
9998219961209	96120996240	DEC 09, 1996	C
9998219961210	96121096241	DEC 10, 1996	C
9998219961211	96121196242	DEC 11, 1996	C
9998219961212	96121296243	DEC 12, 1996	C
9998219961213	96121396244	DEC 13, 1996	C
9998219961214	96121496244	DEC 14, 1996	C
9998219961215	96121596244	DEC 15, 1996	C
9998219961216	96121696245	DEC 16, 1996	C
9998219961217	96121796246	DEC 17, 1996	C
9998219961218	96121896247	DEC 18, 1996	C
9998219961219	96121996248	DEC 19, 1996	C
9998219961220	96122096249	DEC 20, 1996	C
9998219961221	96122196249	DEC 21, 1996	C
9998219961222	96122296249	DEC 22, 1996	C
9998219961223	96122396249	DEC 23, 1996	C
9998219961224	96122496249	DEC 24, 1996	C
9998219961225	96122596249	DEC 25, 1996	C
9998219961226	96122696249	DEC 26, 1996	C
9998219961227	96122796249	DEC 27, 1996	C
9998219961228	96122896249	DEC 28, 1996	C
9998219961229	96122996249	DEC 29, 1996	C
9998219961230	96123096249	DEC 30, 1996	C
9998219961231	96123196249	DEC 31, 1996	C

Exhibit G

```
!               -- Century Conversion --
!
!               Bruce Dickens   Apr 04, 1996 *
!
10    open structure tools:name 'otms_src_dir:tools'
      open #2 : name 'last_inv.dat', access output
         print "    Tools 'Last Inventory Data Format' Check for 1996 Inventory"
         print "ToolNo                ";"  Model No  ";"  LAST_INV ";"LAST_INV "
         print "======                ";"  ========  ";"  ======== ";"======== "
         print "Extract Data:"
         print #2: "ToolNo                ";"  Model No  ";"  LAST_INV ";"LAST_I
NV "
         print #2: "======                ";"  ========  ";"  ======== ";"======
== "
         print #2: "Extract Data:"

20    extract structure tools
         yy$ = lpad$ (element$(tools(last_inv),3,"/"), 2, "0" )
         mm$ = lpad$ (element$(tools(last_inv),1,"/"), 2, "0" )
         dd$ = lpad$ (element$(tools(last_inv),2,"/"), 2, "0" )
         cc$= yy$ + "/" +  mm$ + "/" + dd$
         c1$ = change$(cc$,'/','')
         if c1$[1:2] < '50' then
           c$ = '20' + c1$
         else
           c$= '19' + c1$
         end if
!        include c$ < '19960101'
           sort by tools(model)
           sort by rpad$(c$,8, '0')
!       if c$[1:8] < '19960101'  then
         print tools(toolno); tab(23); tools(model);  &
                      tab(35);tools(last_inv); tab(44); c$
         print #2: tools(toolno); tab(23); tools(model);  &
                      tab(35);tools(last_inv); tab(44); c$
                if valid ( c1$, "digits" ) = 0 then
                print; tab(53); " Date format is not digits"
                print #2:; tab(53); " Date format is not digits"
                end if
!               if valid ( c1$, "minlength 6" ) = 0 then
!               print ;tab(50); " Date format is short"
!               print #2: ;tab(50); " Date format is short"
!               end if
                if tools(last_inv) = "" then
                print ;tab(53); " Date format is blank "
                print #2: ;tab(53); " Date format is blank "
                end if
!       end if
30    end extract
         print
         print "Sorted Data:"
         print
40    for each tools
         c1$ = change$(tools(last_inv),'/','')
         print tools(toolno); tab(23); tools(model); &
          tab(35); tools(last_inv); tab(44); c$
         print #2: tools(toolno); tab(23); tools(model); &
          tab(35); tools(last_inv); tab(44); c$
                if valid ( c1$, "digits" ) = 0 then
                print ;tab(53); " Date format is not digits"
                print #2: ;tab(53); " Date format is not digits"
                end if
!               if valid ( c1$, "minlength 6" ) = 0 then
!               print ;tab(53); " Date format is short"
!               print #2: ;tab(53); " Date format is short"
!               end if
```

Reasons for Allowance

CLAIMS 1-15 ARE PENDING

1. The drawings are objected to on the grounds that FIG 2 does not conform to the claims as amended. In particular, is shows box **36** labeled: reformat data in database. The summary of the invention, the statements at the bottom of page 3 of the Response, and the Reasons for Allowance below specifically preclude this step. This box should be removed.

2. The following is an examiner's statement of reasons for allowance:

The Prior Art of Record, taking into account the Affidavit of the inventor, received 3/24/98, swearing behind the reference of the previous action, does not anticipate nor suggest the set of limitations of the claims, comprising the threshold year digits as used to determine a pair of century digits to be used for computation, but without enlarging the number of date digits of the database.

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

3. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Wayne Amsbury whose telephone number is (703) 305-3828. The examiner can normally be reached on Monday-Thursday from 6:30 AM to 5:00 PM Eastern time.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Thomas G. Black, can be reached on (703) 305-9707. The fax phone number for this Art Unit is (703) 305-9731.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the Group receptionist whose telephone number is (703) 305-9600.

WAYNE AMSBURY
PRIMARY PATENT EXAMINER

April 2, 1998

Shaughnessy

IBM

First Edition, October 1995

(2)

IBM

GC28-1251-00

The Year 2000 and 2-Digit Dates: A Guide for Planning and Implementation

Millennium

The Millennium Journal

A Review of Information Requirements Leading Into Millennium Update

Volume III.III

We are often confronted with questions about storage of date information. Many believe it is possible to reduce the millennium update work effort or minimize storage costs by working around the century. In this Journal we will explore some of the alternatives that we have employed and encountered.

We start this discussion with some reluctance. This stems from our belief that IT organizations should comply with industry and government standards. ANSI, FIPS, and ISO standards all state that the use of YY is permitted only when the date references the current century. SQL also defines date to include the century. For these reasons we have continued to insist that the only correct definition of year is CCYY.

Standards are created to minimize decision making. Where the CCYY standard is not followed, we see the amount of time required to complete the millennium update multiply. Costs often rise to $1.50, and even $3.00, per gross line of code.

Because it is necessary to review and determine how to handle every date calculation and comparison, we have found no significant savings in trying to retain 6-digit dates. Work-arounds cause production cycle times to increase. This requires additional work in performance tuning, and in acquiring new processor resources.

The number of storage options is infinite. We are providing here only a glimpse at the most common ones. The chaos that is caused in trying to keep track of what logic was used, documenting what the values mean, working around mistakes, and training new staff in all these rules costs the organization more in the long run. The variety should itself be a warning to managers to adopt a standard, and enforce it. Since most vendors are not following the standards in their response, the problem of passing date information will be complicated enough.

After all, this Millennium problem...
is about time!

Richard Bergeon
Vice President, Technical Services

The following approaches to dealing with dates are not recommended. Imagine, if you can, an organization in which dates are presented in all the ways mentioned here. This could be your organization. Every format and technique that follows (and more) is being used by vendors as well as internal staff in order to "minimize" the amount of work involved in the update.

Single Digit Century Values

There are situations where space considerations and computational time requirements prohibit full century storage. In those situations, some turn to several single digit century options. A common solution is to store the century as a single digit logic flag.

This is the technique used by IBM's MVS operating system date. Old releases of MVS, CICS used a format 0CYYMMDD. This, and CYYMMDD, are the most common rule used when employing a single digit for the year. MVS 4.2 and CICS (EIBDATE) return values in the form of 0CYYDDD. In all these situations, C = "0" for "19" and "1" equals "20". But, even IBM is not consistent. In the AS/400 world, "0" is preceded the year in 1940 through 1999 and "1" is used for 2000 through 2039.

One of our clients began to use a one digit code for century: "0" = 1800, "1" = 1900, and "2" = 2000. Another variation we have seen is CYYMMDDS which uses signed values. The rule is that 1900 has the value of 0, 1 or +1 represents 2000, and -1 represents 1800. No doubt there are applications which use other numbers and even character values.

Typical storage formats for single digit century are:

Field Description	Format	Field Length[1]
CYYMMDD	BINARY	2
CYYMMDDS	PACKED SIGNED	4
MMDDCYYS	PACKED SIGNED	4
CYYDDD	CHARACTER	6
MM/DD/CYY	CHARACTER	9
CYY/MM/DD	CHARACTER	9

The use of the single-digit century does not save any update work, in fact, its adds to it since some logic must be installed to interpret the code and place the full century in the output fields.

Displaced Dates

Another technique we have seen used is the "displaced date". The year value stored is the difference calculated by subtracting the year from 100. The year 1995 is stored as the value "5". In 2005 the value stored will be "-5".

Some organizations use "9's compliment" - storing the difference the year subtracted from 99. An example, 95 subtract from 99 gives a value of 05. In 2005, 05 is subtracted from 99 giving 94. These techniques require the addition of an arithmetic step every place where the date is validated, stored, retrieved and displayed. Sorts can be tricky. The 9's compliment for the dates 98 through 02 are 02, 01, 99, 98, and 97.

One client adds 30 to the year, truncates the value to two positions, and then compares the value to 30. If the year is "95", the value is 25. Any value less than 30 is assigned a century value of 1900. If the year is "00" the calculated value becomes 30. The century is 2000. This arithmetic process is obviously associated with a logic process. It works a little better for sorts. Use of it becomes difficult if the field contains a date less than "70". It would necessary to use an increment of "40". Sometimes several increment values are used in the same program.

Logic-based Century Determination

Some organizations try desperately to hold on to their 6-digit date standard. Here are two examples of what is done to work around two-digit years. The first, "windowing", is only recommended as temporary measure.

In windowing, the two digit years are left alone in the files. A base year is selected (e.g., "50") where every year starting with (or, in some cases, greater than) "50" through "99" is treated as a 1900 date, and any year less than (or equal to) "50" is treated as 2000.

Sorts require an exit, and different date fields may require a different frame. Again, every place where dates are used in calculations, comparisons or displays, it is necessary to add additional logic. The biggest problem with using windowing is that it will allow all processes to continue to work... even while it produces incorrect results.

There is a logical way of determining the century that always works correctly. It is often possible to derive a logical answer about what century it is based on the contents of another field in the database. Here are two examples:

1. Calculation of current age (current year minus birth year) gives an answer of "09". This could mean 9 or 109 years of age. If a policy charge is high, then one can logically assume that the age is 109. A "relationship" indicator says the person is a "child", then the age is 9.
2. Calculation of mortgage expiration century can be determined by looking at the acquisition date for which the common routine has fixed the century - if the acquisition year is 1994, then the mortgage expiration date must be later (i.e., 44 must be 2044).

Lilian Dates

On October 15, 1582 the Gregorian Calendar was adopted. Its designer was Luigi Lilio - thus the origin of "Lilian". A Lilian clock is a counter incremented by the system at set intervals - usually 100 ms. A subroutine calculates the date and time from the counter using a base date as the zero date.

While some products (e.g., IBM's LE/370) use the Gregorian origin date as the base date, it is not a standard. Use Lilian dates with caution.

1. Aligning Lilian clocks across different products can be tricky.

Common base dates are Jan. 1, 0000, Jan. 1, 1900, Jan. 1, 1964, and Jan. 4, 1980. The IBM MVS base date is 1/1/0000, DB/2's is 1/1/1800, ANSI COBOL reportedly is 1/1/1601, UNIX's clock is 01/01/1970, SAS starts at 01/01/1980, and VMS uses the Smithsonian date - November 17, 1858.

2. Know the limits of the system clock.

The counter field size determines the limit of the calendar - Macintosh's runs out on February 6, 2040. The C (computer language) calendar runs until January 18, 2038.

Satisfying the Standard

Meeting the standard may be possible without expanding data storage. Some organizations have saved space by adopting standard storage formats, eliminating storage formats that take more space.

Field Description	Format	Field Length[1]
YYYYMMDD	PACKED UNSIGNED[2]	4
YYYYDDDS	PACKED SIGNED	4
0YYYYMMDDS	PACKED SIGNED	5
YYYYDDD	CHARACTER	7
YYYYMMDD	CHARACTER	8

Some of our clients have elected to minimize the impact (on data entry procedures, report and screen formats, and century loading procedures) by retaining old formats and extending the year fields. The following formats meet standards, but are awkward to use in comparisons or calculations.

Field Description	Format	Field Length[1]
MMDDYYYY	PACKED UNSIGNED[2]	4
0MMDDYYYYS	PACKED SIGNED	5
MMDDYYYY	CHARACTER	8

Data Dimensions' archives contain over forty different formats that organizations use to store dates. We have only mentioned a few here.

Our message: If you want to simplify the work of the millennium update, stick with the standards.

Trying to save time by adopting an alternative measure only costs more in the long run. There is no significant reduction in either analysis or testing time, and the dollars saved in storage are offset by the costs of additional processing time.

September Issue: Update Strategies

[1] Field lengths are given for IBM mainframes. Other machines my have different word lengths.

[2] Packed unsigned numbers are not available under standard COBOL. Use requires a call to a subroutine available in several date packages.

For additional copies of the Millennium Journal or details on our service offerings please call: 1-800-708-0675, in the U.S. or Canada, or write:

Data Dimensions, Inc.
Corporate Headquarters
777 -108th Ave. NE, Suite 2070
Bellevue, WA 98004
206-688-1000 FAX 206-688-1099
Compuserve 76511,1542, or rbergeon@aol.com

Data Dimensions, Inc. - Mid Atlantic
316 Avalon Drive
Colonial Beach, VA 22443
804-224-2924 FAX 804-224-3001

Data Dimensions, Inc. - Midwest
815 N. Larkin Ave., Suite 104B
Joliet, IL 60435
815-744-8006 FAX 815-744-5113

Data Dimensions, Inc. - Northeast
304 Boston Post Road
Wayland, MA 01778
508-358-8010 FAX 508-358-8009

Data Dimensions, Inc. - Southeast
2849 Executive Drive, Suite 230
Clearwater, FL 34622
813-573-2030 FAX 813-573-9466

Data Dimensions, Inc. - Southwest
15770 N. Dallas Pkwy.
Suite 600, LB46
Dallas, TX 75248
214-387-7442 FAX 214-387-7441

Data Dimensions, Inc. - West
100 South Ellsworth, 9th Floor
San Mateo, CA 94401
415-696-3148 FAX 415-348-3017

Data Dimensions, Inc. - Eastern Canada
365 Ontario Street
Toronto, Ontario M5A 2V8
Phone/FAX 416-968-3331

Millennium (UK) Ltd.
Rowley House, 5 Hosiers Lane,
The Quay, Poole, Dorset, BH15 1HL
01202 667557 FAX 01202 660979

Millennium Scandinavia
Wavulinintie 3
00210 Helsinki, Finland
358-0-615181 FAX 358-0-692-2663

Ohms

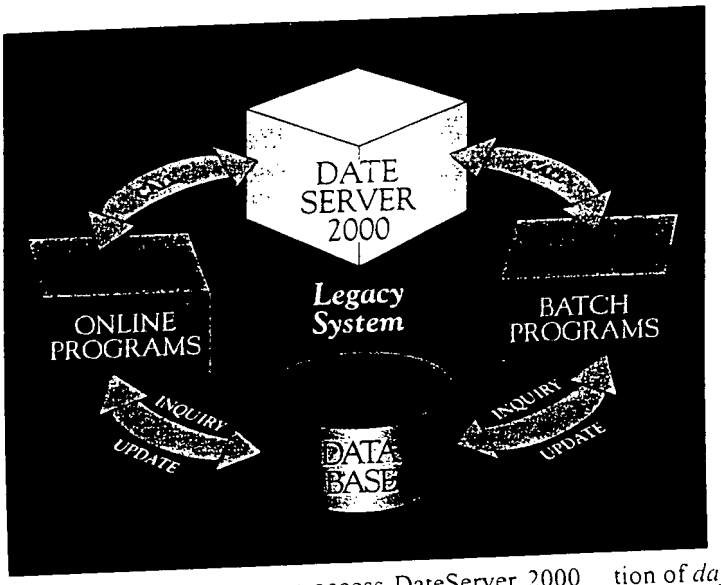
Roberts

FIRST IMPRESSION

DateServer™ 2000

Computer Software Corp. Prepares Legacy Systems For Year 2000

For many organizations, preparing legacy systems to correctly process multicentury dates will likely require a major expenditure of time and money over the next few years. Changes must be made to all application systems that make date field comparisons or calculations using date field values with only a two-digit representation for the year. Computer Software Corp.'s DateServer 2000 software offers a unique solution to significantly reduce the time and effort to prepare for processing in the year 2000 and beyond.



The conventional approach to the year 2000 problem is to expand date field sizes to provide for date values with a four-digit year. This file-conversion, or "sledgehammer," approach is a costly solution for most enterprises. It requires special programs to convert all date fields within all databases. Changes to all existing input screens and files are also required. It requires coding changes to all current programs that process the new input dates and the converted databases as well as changes to all output screens and reports. Finally, this conventional approach requires the testing of all the new and changed programs, and a great deal of coordination during the actual system conversion and implementation.

No Database Conversions

In contrast, the DateServer 2000 solution eliminates the requirement to convert any files or databases. Using the familiar CALL statement interface, programmers can access DateServer 2000 routines to make date field comparisons or calculations on all date fields in their present database formats. During program execution, DateServer 2000 routines will assign the appropriate century value, based on the system's current date and DateServer 2000 installation parameters, to correctly perform requested date comparisons or calculations.

One Program At A Time

Eliminating the need for file or database conversions and the requirement to expand input or output date formats, the DateServer 2000 solution simplifies the preparation and coordination efforts required to process multicentury dates. Application programs may be changed, tested and implemented one program at a time without worry about complex database interaction with other on-line or batch programs. Rather than having to fit a major database conversion into a busy project schedule, these application program changes could be merged with other project testing.

DateServer 2000 software provides a comprehensive, efficient and convenient set of routines for date comparison, calculation and or manipulation. The routines support 36 date formats including character, packed and binary Gregorian and Julian dates, and *day of week* and *relative day* values for all dates in the Gregorian calendar. The routines perform: date validation; date comparison; conversion from one format to another; determination of *day of week* and *relative day* values; calculation of the difference between any two dates in number of years and/or days; and determination of the date resulting from the addition or subtraction of any number of days to or from a given date.

Even enterprises that choose the file conversion approach for some legacy systems could still use the DateServe 2000 solution for other systems.

Computer Software Corp. offers versions of the DateServer 2000 software to operate in either an IBM MVS or IBM VSE environment. A CICS demonstration program is also included for CICS users. Priced on a tiered basis with multiple-site discounts, DateServer 200 software is available starting at $10,000. For more information, contact Compute Software Corp., 19100 Detroit Road Cleveland, OH 44116, (800) 908-200 Fax (216) 333-8288, or (216) 333-908 outside the United States.

— Chris Roher

DeJager

DOOMSDAY

The date change in the year 2000 — an event that may trigger fatal errors in mission-critical systems — is only 2,308 days away. Many IS people are unprepared or unconcerned.

BY PETER DE JAGER

Have you ever been in a car accident? Time seems to slow down as you realize you're going to crash into the car ahead of you.

It's too late to avoid it — you're going to crash. All you can do now is watch it happen.

The information systems community is heading toward an event more devastating than a car crash. We are heading toward the year 2000. We are heading toward a failure of our standard date format: MM/DD/YY.

Unfortunately, unlike the car crash, time will not slow down for us. If anything, we're accelerating toward disaster.

THE COST FOR PROGRAMMING TO ADJUST ALL SYSTEMS FOR THE YEAR 2000: $50 BILLION.

This is a good news/bad news story. First the bad news: There is very little good news. There is no way to avoid the fact that our information systems are based on a faulty standard that will cost the worldwide computer community billions of dollars in programming effort.

Perhaps more importantly, we are going to suffer a credibility crisis. We and our computers were supposed to make life easier; this was our promise. What we have delivered is a catastrophe.

The problem is twofold: the date issue itself and, more importantly, our reluctance to address the problem.

THE COST WHEN YOU ADD DESIGN, MANAGEMENT, HARDWARE, SOFTWARE AND SUPPORT: $75 BILLION.

Source: Cost and programmer time estimates come from Bill Goodwin, editor and publisher of FYI, Inc.'s newsletter for people concerned about the impact of the year 2000

Problem ID

What exactly is the "problem"? To save storage space — and perhaps reduce the amount of keystrokes necessary to enter a year — most IS groups have allocated two digits to the year. For example, "1993" is stored as "93" in our data files, and "2000" will be stored as "00." These two-digit dates exist on millions of data files used as input to millions of applications.

This two-digit date affects data manipulation, primarily subtractions and comparisons. For instance, I was born in 1955. If I ask the computer to calculate how old I am today, it subtracts 55 from 93 and announces that I'm 38.

So far so good. But what happens in the year 2000? The computer will subtract 55 from 00 and will state that I am -55 years

Doomsday, *page 108*

TOOLS FOR 2000

CALENDAR ROUTINES
TRANSCENTURY DATA SYSTEMS
(415) 255-7651
A single set of call routines with a 10,000-year date range (from 1AD to 9999 A.D.). Data offers different date formats, 10 different holiday tables and 2,400 definitions of the workweek.

Package can accept dates without centuries by defaulting to the century based on the cutoff year of a company's choice.

GENERAL PURPOSE SYSTEMS ANALYZER (GPSA)
COBOL MAINTENANCE TECHNOLOGIES
P.O. BOX 122069
CHULA VISTA, CALIF 91912-6769
Helps standardize data areas and rationalize processing routines in mainframe Cobol programs. Basically, IS downloads a Cobol program onto a PC and analyzes it using GPSA.

GPSA lists all data names that appear to contain dates. The data definitions for each data name are listed and will help determine whether the year is stored in a two- or four-digit field. The system also offers users a list of all procedure code referring to the list of data names. Users can see whether there are date compares that will cause problems or any data names that were not on the initial list.

PM/SS
ADPAC CORP.
(415) 974-6699
The product reportedly helps identify all date occurrences and their primary and secondary relationships. It is said to locate affected data sets, databases, files, data name definitions and data storage locations, as well as data name use at the line-of-code level.

SE/ONE
SOFTWARE ECLECTICS, INC.
(404) 667-9117
Billed as a "re-engineering" tool for Cobol and CICS. Version 3.2 has a D-Ray feature that gives users the capability to highlight date operations within a program. This is said to enable maintenance modifications to accommodate the century change.

TICTOC
ISOGON CORP.
(212) 967-2424
Date and time testing tool for MVS/ESA and XA that claims to support all languages. It lets users test jobs using different dates without requiring a separate test system and without affecting other jobs running at the same time.

Doomsday, *from page 105*

old. This error will affect any calculation that produces or uses time spans, such as an interest calculation.

If you have some data records and want to sort them by date (e.g., 1965, 1905, 1968), the resulting sequence would be 1905, 1965, 1966. However, if you add in a date record such as 2015, the computer, which reads only the last two digits of the date, sees 05, 15, 65, 66 and sorts them incorrectly.

These are just two types of calculations that are going to produce garbage. There are others.

The task facing us is to identify and correct all the date data and check the integrity of all calculations involving date information. We must correct the data residing in all data files or write code to handle the problem.

AVERAGE TIME NEEDED FOR CODE 2000:
7 DAYS PER PROGRAM.

The starting point

How do we identify the problem data and the associated calculations? We have few, if any, standards for labeling data used in date calculations. The only choice we have is to examine each line of code and make the necessary changes.

One IS person I know of performed an internal survey and came up with the following results: Of 104 systems, 18 would fail in the year 2000. These 18 mission-critical systems were made up of 8,174 programs and data-entry screens as well as some 3,313 databases. With less than seven years to go, someone is going to be working overtime.

By the way, this initial survey required 10 weeks of effort. Ten weeks just to identify the problem areas.

How many systems do you have? How many lines of code do you have in your organization? How many data files? How many maintenance programmers?

The problem extends beyond mere calculations and into the I/O processes of every application. Can you enter 2000 into your data screen, or can you enter only two digits, forcing the input of 00? Can your hard-copy reports print four digits?

The crisis is very real and potentially very costly. Ken Orr, principal at the Ken Orr Institute, and Larry Martin, president of Data Dimensions, Inc., estimate that Fortune 50 organizations will each have to spend about 35 to 40 cents per line of code to convert all their existing systems to accept the change from the year 1999 to 2000.

TOTAL PROGRAMMING TIME FOR ALL SYSTEMS:
1.2 MILLION MAN-YEARS.

That translates into about $50 million to $100 million for each company. The mind boggles at a maintenance problem with that price tag.

And the costs could be even higher. "The truth is, until we work through a complete cycle with some large organization, we are not going to really know," Orr says.

I have spoken at association meetings and seminars, and when I ask for a show of hands of people addressing the problem, the response is underwhelming. If I get one in 10 respondents, I'm facing an enlightened group.

Typically, all I get are snickers and comments such as, "I won't be in this position or this company in the year 2000. It's not my problem."

This attitude in the computing community is the real problem. It is very difficult for us to acknowledge that we made a "little" error that will cost companies millions of dollars. It is also a "pay me now or pay me later" situation.

"We in the IS industry have not been paying our way," says Gerald Weinberg, author of *Quality Software Management* and winner of the 1991 J. D. Warnier Prize for Excellence in Information Science. "We have been building up a 'national debt' just as surely as the U.S. has been building up a money debt. It will be paid by our children — our successors — one way or another," Weinberg says.

We don't have a choice. We must start addressing the problem today or there won't be enough time to solve it. Status quo means applications that will produce meaningless results in the new millennium.

Weinberg says he believes this procrastination is an indication of deep management malaise. "If software engineering managers cannot manage a change that they've had 1,000 years to prepare for, how can we expect them to manage a change that happens without notice? In other words, if this change causes a crisis in your organization, *everything* will cause a crisis in your organization — and often *nothing*

Bearings firm goes on offense

JUST DO IT

BY LORY ZOTTOLA DIX

Torrington Co. doesn't have time for seers predicting dire consequences when the final tick of the clock strikes a change to the year 2000. That's because this bearings manufacturing company, a division of Ingersoll-Rand Co., is too busy doing something about it.

In 1991, at the suggestion of an employee and as part of its drive toward total quality management, Torrington convened a seven-member team, headed by programmer/analyst Bob Hartman-Berrier, to tackle the century date change issue. The Torrington, Conn.-based company knew that because certain of its programs stored dates by their final two digits, a changeover to the year 2000 could mean fatal errors that might throw its global systems into an uproar.

The group brainstormed about problems areas — the company has a mix of mainframes, minicomputers, PCs and local-area networks — and interviewed businesspeople throughout the firm.

Members took an inventory of in-house and outside software products. They sent a questionnaire to Torrington's 90 vendors (30 mainframe and minicomputer, 60 PC) to gauge product support for four-digit years.

Vendor responses were encouraging. Of the 98% mainframe and 60% PC vendors that returned the survey, nearly all of them either could accommodate the century date or had fixes in the works.

The in-house inventory showed that 30% of Torrington's programs and 25% of its files and databases required changes, Hartman-Berrier says. At 25 hours per program and 40 hours per database file to reformat, unload and reload, the team

Here's what's on Torrington's to-do list to prepare for 2000:

1. Set standards. All in-house and vendor system files and any date fields will have a four-digit year. (Status: Done.)

2. Accommodate legacy systems. Despite its move to a distributed environment, the company will maintain its mainframes. It is actively seeking a mainframe compiler that supports four-digit years, something its current compiler doesn't do. (Status: In progress.)

3. Investigate and buy a data analysis tool to identify location of date fields within programs. (Status: Will be complete in 12 to 18 months.)

4. Develop bridge modules. Some programs, especially those dealing with forecasting, will need to work with the century date as early as 1996. These "critical" systems (such as payroll and job scheduling) will need bridge modules to handle century rollover. These software fixes are a less time-intensive alternative to changing native code. (Status: Will be complete in 12 to 18 months.)

5. Identify critical systems needing bridge modules and prioritize them. (Status: In progress.)

6. Do it. Put the recommendations in action.

will cause a crisis."

The inability of the industry to even think about such a project is troublesome. "No one wants to step up to the issue — not [IS] management, not the vendors, not the industry gurus," Orr says. "As with all legacy systems, this problem is messy, expensive and unromantic. No one wants to go in and tell management they have a multimillion-dollar requirement just to keep the business running and that they really have no options."

The reason nothing is being done, says Capers Jones, chairman at Software Productivity Research, Inc., is that the software industry isn't used to taking long-term preventative steps. "I expect that most companies will not start worrying about the problem until 1999," Jones says. "For some, this will be too late."

Now the good news

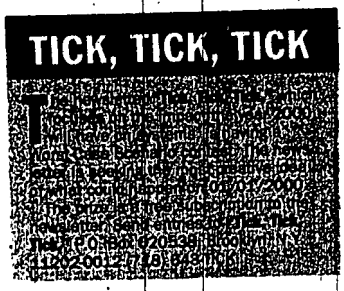
There is good news. Object-oriented systems may be able to help. Faced with the huge maintenance costs of fixing their systems, firms may opt to rewrite systems from scratch using object-oriented programming techniques. Tom Love, IBM vice president of the Object-Oriented Group, is a proponent of this theory.

Some companies are unveiling testing and inventory tools that may ease the identification of trouble spots.

Others are hoping that bombarding people with information is the best remedy. To that end, William Goodwin in Brooklyn, N.Y., publishes a newsletter entitled "Tick, Tick, Tick," which brings together people in the IS industry concerned about the impact of the year 2000.

But is the warning falling on deaf ears?

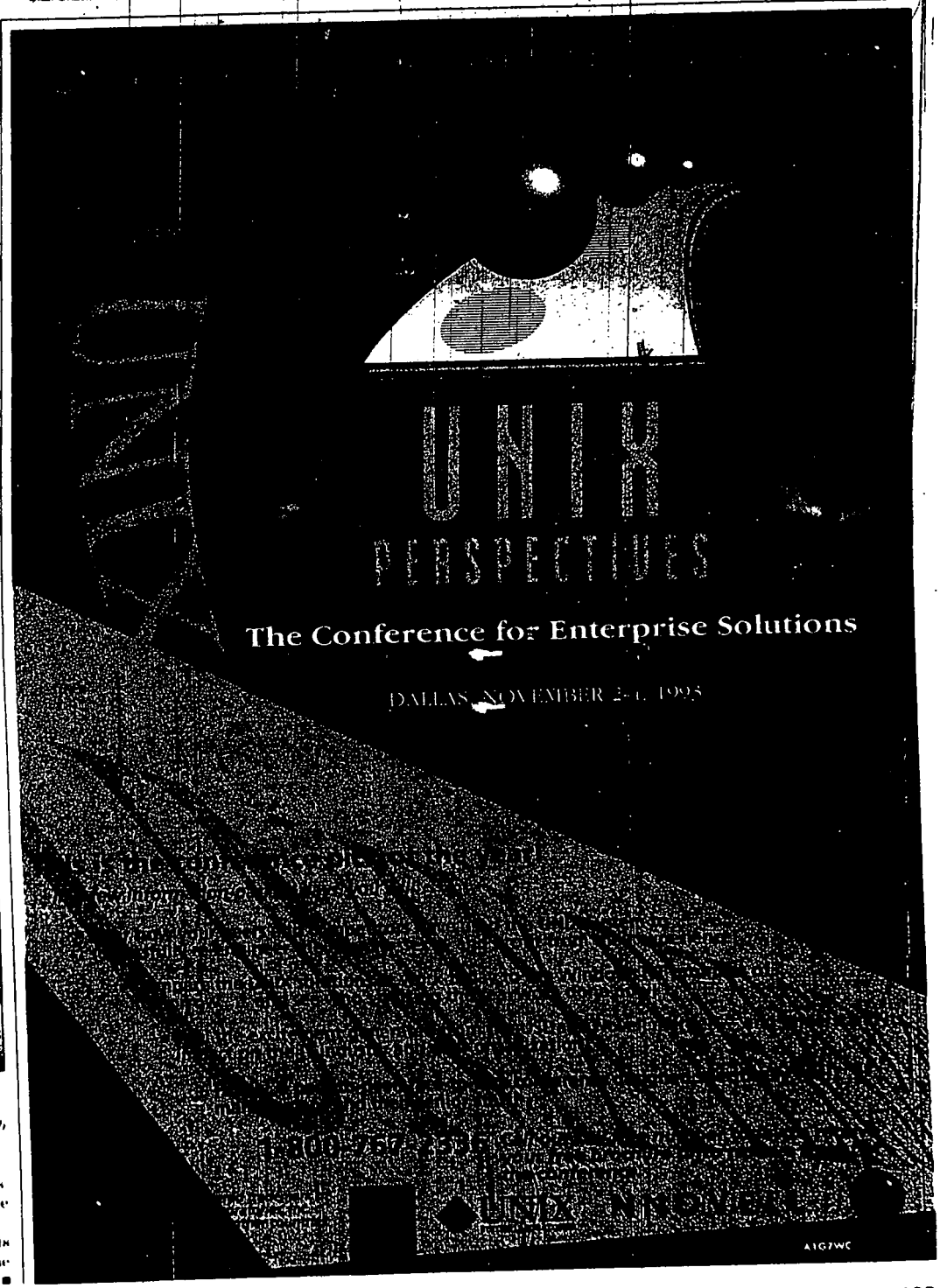
TICK, TICK, TICK



"I feel like a lone voice crying in the wilderness," says Brian Pitts, one of Goodwin's subscribers and project manager at Berry Co. in Dayton, Ohio. "Current economic conditions are making this problem more difficult to address. Management is focused on short-term results and is placing long-term negative consequences on the back burner."

The next seven years will be filled with dire predictions. "You are going to become very, very tired of millennium moaners telling you that your software will fail as it enters the new millennium," says Nicholas Zvegintzov, publisher of *Software Maintenance News*. "But be patient with them. There really is something to be said for them." ■

De Jager is an industry speaker on the topics of change, creativity and management of technology. He can be reached at (416) 792-8706 or via CompuServe (70011,2576) and MCI Mail (PDEJAGER).



expects the work load to be heavy.

Costs have been a little trickier to pinpoint. While Hartman-Berrier says worst-case costs to revamp systems could reach $3.5 million, the best-case could be one-tenth that figure.

Hartman-Berrier and Ken Even, manager of corporate support, hesitate to commit to a number on project cost because of the firm's move to a distributed setup. "Distributed systems may mean we remove some applications, keep others. If we remove a system, we don't have to change it, reducing our labor efforts

The Torrington 2000 team: *(front) Traci Winegar, Alison Anstett, Evelyn Palazzini, (back) John Draghi, Bill Beyer, Bob Bouchard, Hartman-Berrier*

and resources," Hartman-Berrier says. "We can take a felt-tip marker and strike programs and files from our list."

His advice to other IS professionals is to "plan, plan, plan and act, act, act." The year 2000 is coming ■

Common Lisp, The Language (1984)

25.1.4.1 Decoded Time

A *decoded time* is an ordered series of nine values that, taken together, represent a point in calendar time (ignoring *leap seconds*):

Second

An *integer* between 0 and 59, inclusive.

Minute

An *integer* between 0 and 59, inclusive.

Hour

An *integer* between 0 and 23, inclusive.

Date

An *integer* between 1 and 31, inclusive (the upper limit actually depends on the month and year, of course).

Month

An *integer* between 1 and 12, inclusive; 1 means January, 2 means February, and so on; 12 means December.

Year

An *integer* indicating the year A.D. However, if this *integer* is between 0 and 99, the ``obvious'' year is used; more precisely, that year is assumed that is equal to the *integer* modulo 100 and within fifty years of the current year (inclusive backwards and exclusive forwards). Thus, in the year 1978, year 28 is 1928 but year 27 is 2027. (Functions that return time in this format always return a full year number.)

Day of week

An *integer* between 0 and 6, inclusive; 0 means Monday, 1 means Tuesday, and so on; 6 means Sunday.

Daylight saving time flag

A *generalized boolean* that, if *true*, indicates that daylight saving time is in effect.

Time zone

A *time zone*.

The next figure shows *defined names* relating to *decoded time*.

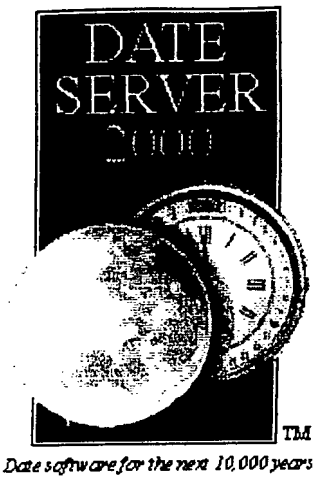
`decode-universal-time` `get-decoded-time`

Figure 25-5. Defined names involving time in Decoded Time.

Starting points | Contents..... Chapter 1..... Chapter 2..... | Master Index M N | Symbol Index S T | Glossary, n. Index of terms. | x3j13 issues

14

Computer Software Corp.
Web page on DateServer 2000



DATE SERVER™ 2000 FASTEST DATE PROCESSING ROUTINES

for mainframe legacy application systems using a

Windowing or Expansion Solution.

Computer Software Corporation's DateServer 2000 software is currently called millions of times each day. It has been solving multi-century date processing problems for COBOL and Assembler legacy application systems since January, 1994.

---

without expensive date field expansion and database conversions.

without expensive changes to input screens, database fields, reports, etc.

The DateServer 2000 solution is about . It's about preparing your legacy systems for a in the future, for all in the future. And it's about saving a significant amount of MIS in the process. Our solution can deliver these benefits today, in a fraction of the MIS that other multi-century approaches will require.

Review DateServer™ 2000 Benefits

Review DateServer™ 2000 Features

Review DateServer™ 2000 Date Formats

Review DateServer™ 2000 Performance Data

ELIMINATE TIME CONSUMING DATA BASE CONVERSIONS.

Many legacy systems, comprised of both online and batch sub-systems, consist of hundreds of programs, thousands of modules, and millions of lines of code. Data files in a legacy system data base may be interactively used by hundreds of programs. The conventional approach to providing multi-century date capabilities requires the conversion of date formats in all data bases in each legacy system. This means new file conversion programs, increased application program changes, complex system testing, and a highly-coordinated data base conversion and software implementation. It will likely take ***years*** of MIS time to convert some legacy systems using a conventional approach.

Enter The New Century With Existing Data Bases!

Using the DateServer 2000 solution*, ***no data base conversions*** are needed! All of your current legacy system's data base date formats can remain as they are until you decide (if you ***ever*** decide) to convert them to a different format. This means that the date formats now used in internal files, reports, screens, input cards, and imported files can remain unchanged. Since no data base conversions are needed, DateServer 2000 saves your MIS team a significant amount of time and resources that your company can invest into other strategic projects.

It's About Time For Date Server™ 2000

- **Easy To Use**

DateServer 2000 software is accessed using the familiar CALL statement interface. It supports 36 different date formats, performs date comparisons, date validation, date arithmetic, and provides day-of-week functions.

- **Most Comprehensive**

DateServer 2000 provides a comprehensive set of routines to accomplish any of the functions required for multi-century date processing. This includes the processing of invalid date values used for a special purpose, for example: all zeros or all nines in a date field indicating a date value lower or higher than any valid date.

- **Most Efficient**

DateServer 2000 routines are written in assembler to achieve maximum processing efficiency. In addition, each function is requested individually to eliminate any processing overhead that is not required to accomplish the requested function, for example: the validation of each date field is not required to perform the comparison of two date fields.

- **Easy to Install**

DateServer 2000 requires an IBM® MVS, IBM VSE or equivalent environment. The software comes on a single diskette and can be installed in about one hour. A DateServer 2000 CICS demonstration program is included, and a free trial period is offered.

- **Relax on Friday, December 31, 1999**

You'll be able to enjoy your evening because you know that all of your legacy systems are 100% ready for the year 2000 and beyond - and have been for years!

COMPUTER SOFTWARE CORPORATION

has been in operation since 1980. The company's founders have been providing mainframe computer solutions since 1965 and microcomputer solutions since 1978. Computer Software Corporation has developed specialized mainframe software products for the banking industry and specialized PC data collection software for the manufacturing and distribution industries.

Computer Software Corporation
24864 Detroit Road
Cleveland, Ohio 44145
1-800-908-2000
FAX (440) 808-8086
(440) 333-4420 outside the United States

Visit our Home Page at www.dateserver.com

E-Mail Computer Software Corporation at CSC@DateServer.com

Return to ___ - Benefits - Features - Date Formats - Performance Data

You are guest 03177 to visit DateServer™ 2000's Page since 11/25/96.

---

* U.S. Patent No. 5,630,118
IBM is a registered trademark of International Business Machines Corporation.
DateServer is a trademark or a registered trademark of 2000, Inc.

ISO 2014 Numeric Calendar Dates
1976

Archival Moving Image Materials

Writing of calendar dates in all-numeric form (Appendix C: ISO Date Standard)

0 Introduction:

In all forms of international traffic and exchange, dates must be clearly designated and able to be compared without any ambiguity.

The International Standard for writing of calendar dates in all numeric form has been prepared to obviate the confusion arising from misinterpretation of the significance of the numerals in a date written with numerals only; it is considered that similar confusion does not arise when the month is spelled out, either in full or in abbreviated form.

The occasions on which an all numeric date might be used have been examined and the advantages for these occasions of the descending order year - month - day have been found to outweigh those for the ascending order day - month - year, established in many parts of the world.

The advantages of the descending order include the following in particular:

- the ease with which the whole date may be treated as a single number for the purpose of filing and classification (for example for insurance and social security systems).
- arithmetic calculation, particularly in some computer uses.
- the possibility of continuing the order by adding digits for hour - minute - second.

1 Scope:

The International Standard specifies the writing of dated of the Gregorian calendar in all numeric form, signified by the elements year, month, day.

2 Field of Application:

The International Standard is applicable whenever a calendar date containing the elements year, month, day is written in all numeric form.

3 Rules for Writing calendar Dates:

3.1 Sequences:

An all numeric date shall be written in the following order:

- year - month - day

3.2 Characters:

An all numeric date shall be expressed exclusively in arabic numerals, i.e. by using only the decimal digits 0,1,2,...,9.

3.3 Elements:

An all numeric date shall consist of:

- four digits to represent the year

Note: Two digits may be used when no possible confusion can arise from the omission of the century, however, four digits should be applied especially in correspondence and for documentation purposes to indicate clearly that the descending order is used.

- two digits to represent the month.

- two digits to represent the day.

3.4 Separator:

Where a separator is used in an all-numeric date, only a hyphen or a space shall be used between year and month, and between month and day.

3.5 Examples:

The 1st of July 1976 shall be written in one of the following ways:

a) 19760701

b) 1976-07-01

c) 1976 07 01

See also:

Appendix C: ISO Date Standard

Appendix C: ISO Date Standard

INTERNATIONAL STANDARD ISO 2014

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • ORGANISATION INTERNATIONALE DE NORMALISATION

Writing of calendar dates in all-numeric form

Representation numerique des dates

First Edition -- 1976-04--01

UDC 529.2 : 003.35	Ref. No. ISO 2014-1976 (E)
Descriptors : calendar dates, writing, numeric representation	

Forward:

ISO (the International Organization for Standardization) is a worldwide federation of national standards institutes (ISO Member Bodies). The work of developing International Standards is carried out through ISO Technical Committees. Every Member Body interested in a subject for which a Technical Committee has been set up has the right to be represented on the Committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work.

Draft International Standards adopted by the Technical Committee are circulated to the Member Bodies for approval before their acceptance as International Standards by the ISO Council.

Prior to 1972, the results of the work of the Technical Committees were published as ISO Recommendations; these documents are not in the process of being transformed into International Standards. As part of this process, Technical Committee ISO/TC 154 has reviewed ISO Recommendations R 2014 and found it technically suitable for transformation International Standards ISO 2014 therefore replaces ISO Recommendation R 2014 - 1971 to which it is technically identical.

ISO Recommendation R 2014 was approved by the Member Bodies of the following countries:

Austria	Italy	Sri Lanka
Belgium	Japan	Sweden
Canada	Korea,, Dem P. Rep of	Switzerland
Egypt, Arab Rep. of	Korea, Rep. of	Thailand
France	Netherlands	United Kingdom
Germany	Poland	U.S.A.
Greece	Portugal	Yugoslavia
Hungary	South Africa, Rep. of	
India	Spain	

The Member Bodies of the following countries expressed disapproval of the Recommendations on technical grounds:

- Czechoslovakia
- Iraq
- Ireland

No Member Body disapproved the transformation of ISO/R 2014 into an International Standard.

For body of ISO Date Standard, see:

Writing of Calendar dates in all-numeric form

See also:

Archival Moving Image Materials: Contents